A Continuous Time Markov Chain framework for Insertion Language Models

Dhruvesh Patel[♢] Benjamin Rozonoyer[♢] Soumitra Das[♢]
Tahira Naseem[♡] Tim G. J. Rudner[♠] Andrew McCallum[♢] [1]

[1] [♢]UMass Amherst [♡]IBM Research [♠]New York University

Abstract

Sequence generation through insertion of tokens offers several advantages over left-to-right generation and mask-based generation—especially for planning tasks that require look-ahead, and for tasks that need to satisfy relative ordering constraints. Existing formulations of insertion-based generation have largely been ad-hoc. In this paper, we derive a diffusion-style denoising objective from first principles by formulating the noising process as a continuous-time Markov chain—defined over the space of variable-length sequences—that drops tokens uniformly with a time-dependent rate. We show that, with certain approximations, previous formulations of insertion language models can be viewed as special cases of this denoising framework. We propose new network parameterizations that explicitly model the rate matrix of the generative Markov chain, leading to principled sampling procedures. 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.

1 Introduction

Auto-regressive language models (ARMs) generate sequences by predicting tokens from left-to-right. They can naturally generate variable length sequences by using 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 restricts the sampling to block-based left-to-right sampling [Nie et al., 2025]. 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 of benefits of ARMs and MDMs — they naturally generative 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 certain 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 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 as well as 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 a bound on the data log-likelihood for the parameterized generative process, and show that by using some reasonable approximations, one can learn the parameters of both the parameterizations efficiently by minimizing the bound.

The main contributions of this work are as follows:

1. We derive a principled diffusion-style denoising objective and propose two parameterizations of the rate matrix of the generative process.
2. We show that the resulting sampling procedures unify the existing formulations of Insertion Language Models.
3. We validate the proposed approach on synthetic planning tasks and language modeling tasks.

2 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 pre-training 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 4.3.2). 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.

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. Havasi et al. [2025] uses conditional discrete flow matching with Bregman Divergence to train a Markov chain that generates by performing *edit* operations on the sequence. We note that our training objective is quite similar, but is derived from the perspective of diffusion.

3 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 Section 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., $[\![n]\!] = \{1, 2, \dots, n\}$. 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 $P\{X_s = y \mid X_t = x\}$, and in case of discrete time processes, we will use $\underline{p}_{r|k}(y \mid x)$.

Continuous-Time Markov Chains We briefly discuss continuous-time Markov chains (CTMCs), their characterization in terms of the rate matrix, and their relationship to discrete time Markov chains. In Section 4, we will leverage the relationship between DTMC and CTMC 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, with $\underline{T}_n$ denoting the time of the n-th transition, and let $\underline{X}_n$ be a discrete-time process defined as $\underline{X}_n = X_{\underline{T}_n}$. * Then $\underline{X}_n$ forms a discrete-time Markov chain (DTMC). Furthermore, the transition rate matrix of X_t can be described using the transition rate parameter $\lambda_t(x)$ and the transition kernel $K_t(x, y) \geq 0$ of the embedded DTMC with no self-transitions:

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

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

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

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 $\lambda_t(x) = \sum_{y \neq x} R_t(x, y)$.

The task of path finding on variable length star graphs [Patel et al., 2025] creates such a condition.

4 Method

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.

4.1 Noising Process

The noising process proceeds by deleting tokens from the ground-truth sequence $\boldsymbol{x}_0$. We will 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

*By convention, $\underline{T}_0 = 0$, and $\underline{X}_n = \underline{X}_{n-1}$ when $\underline{T}_n = \infty$ to handle the case of absorbing states. See Appendix B for details.

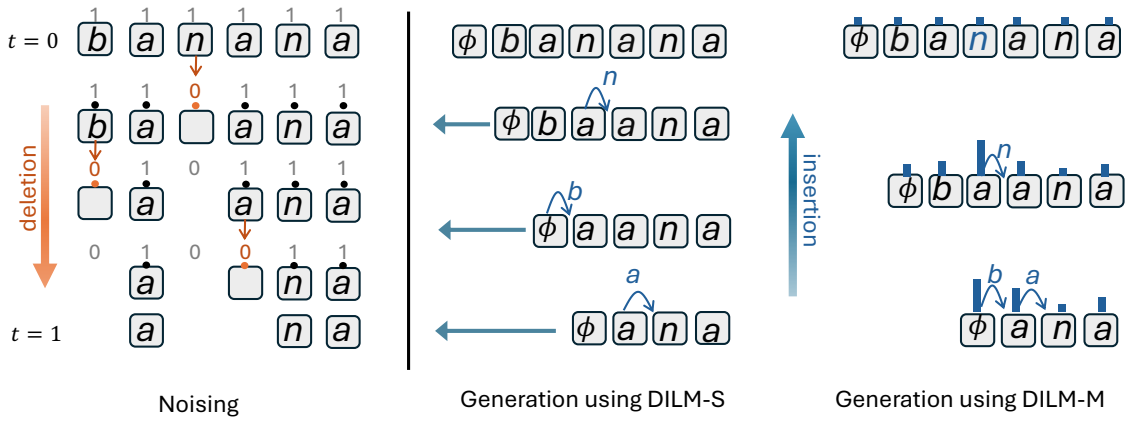

Figure 1: The noising process, shown on the left, is a continuous time Markov chain that deletes one token at a time uniformly with the deletion events arriving with rate σ_t. The bit vectors show the deletion path w.r.t. to the original sequence $\boldsymbol{x}_0 = (6, \text{banana})$. On the right we show two parameterization 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 use 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.

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 $\text{Idx}(\boldsymbol{b}) \coloneqq \{i \in [\![n]\!] \mid b^i = 1\}$ be the set of indices of the ones in $\boldsymbol{b}$, and $\boldsymbol{x}[\boldsymbol{b}] \coloneqq \boldsymbol{x}^{\text{Idx}(\boldsymbol{b})}$ 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 1. 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

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

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 the following one-step transition kernel and transition probability,*

respectively:

$$K((n,\dot{\boldsymbol{x}}),(m,\dot{\boldsymbol{y}})) = \frac{1}{n}\sum_{\boldsymbol{b}\in\mathbb{B}_{n,m}} \delta_{n-1}(m)\,\delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}),$$
$$\underline{p}_{k+r|k}\left((m,\dot{\boldsymbol{y}}) \mid (n,\dot{\boldsymbol{x}})\right) = \hat{C}^n_m\,\delta_r(n-m)\sum_{\boldsymbol{b}\in\mathbb{B}_{n,m}} \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}).$$

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 $\hat{C}^n_m = \frac{(n-m)!}{n(n-1)\ldots(m+1)} = \frac{1}{\binom{n}{n-m}}$. The formal proof is given in Appendix C.

Remark. 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 are more than one masks $\boldsymbol{b} \in \mathbb{B}_{n,n-r}$ 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 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 are now ready to give a complete description of the continuous time deletion process. Intuitively, the $\underline{\boldsymbol{X}}_k$ is converted into a continuous time $\boldsymbol{X}_t$ by dispersing the deletion steps over the time interval $[0,1]$ with deletion rate $\lambda_t(\boldsymbol{x})$ per unit time. The following is a heuristic description of how the process evolves:

$$\boldsymbol{X}_{t+dt} = \begin{cases} \boldsymbol{X}_t & \text{with prob. } 1-\lambda_t(\boldsymbol{X}_t)\,dt \\ \boldsymbol{Y}\sim K(\boldsymbol{X}_t,\boldsymbol{Y}) & \text{with prob. } \lambda_t(\boldsymbol{X}_t)\,dt. \end{cases} \tag{2}$$

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 $\lambda_t((r,\dot{\boldsymbol{x}})) = \sigma_t\delta\{r>0\}$, where $\sigma:[0,T]\to\mathbb{R}_+$ is a scalar noise schedule. The following proposition

Proposition 1 (Transition Probability)**.** *Let $\boldsymbol{X}_t$ be a continuous time stochastic process described in equation 2, with transition rate $\lambda_t((r,\dot{\boldsymbol{x}})) = \sigma_t\delta\{r>0\}$ 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*

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

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

Proof 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 1 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 the noising process is described in Algorithm 1.

Algorithm 1 Sampling from the Noising Process

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

4.2 Generative Process

The time reversal of a continuous time Markov chain on a finite horizon is also a continuous time Markov chain with its rate matrix given by $\hat{R}_t(x, y) = \frac{p_t(y)}{p_t(x)} R_t(y, x)$. [†] In our case, the time reversal is the generative process that constructs a sequence by inserting tokens. Let $\mathbb{X}_n$ denote $\{n\} \times \mathbb{V}^n$, $\text{ins} : \mathbb{X}_n \times [\![n+1]\!] \times \mathbb{V} \to \mathbb{X}_{n+1}$ 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 $\hat{R}_t^\theta$.

Proposition 2. *The log-likelihood of the data $\log p_{\text{data}}(x_0)$ under the parameterized time reversal of the noising process which is a CTMC on a finite state space is bounded from below by*

$$\mathbb{E} \sum_{y \neq x} \left[-\hat{R}_t^\theta(x, y) + \frac{p_{t|0}(y|x_0)}{p_{t|0}(x|x_0)} R_t(y, x) \log \hat{R}_t^\theta(x, y) \right] + C, \tag{3}$$

where the expectation is over $t \sim$ Uniform$[0, 1]$, $x \sim p_{t|0}(x \mid x_0)$, and C is independent of θ.

The proof presented in Appendix C.2.

4.3 Parameterizations

In this section, we discuss different options for parameterizing the rate matrix of the generative process, and their relation to some existing formulations of insertion language models.

4.3.1 Joint Probability and Length-to-go

Let $\hat{p}_t^\theta(i, w \mid \boldsymbol{x})$ denote the parameterized probability of $\text{ins}(\boldsymbol{x}, i, w)$. We can parameterize the the rate matrix of the generative process as

$$\hat{R}_t^\theta(\boldsymbol{x}, \boldsymbol{y}) = \hat{\lambda}_t^\theta(\boldsymbol{x}) \sum_{i=1}^{m+1} \sum_{w \in \mathbb{V}} \hat{p}_t^\theta(i, w \mid \boldsymbol{x})\, \delta_{\text{ins}(\boldsymbol{x}, i, w)}(\boldsymbol{y}). \tag{4}$$

We plug this parameterization into Equation (3), and make the following two approximations resulting in the Corollary 1 below (the full derivation is given in Appendix C.2).

1. When converting the summation over $\boldsymbol{y}$ into an expectation over allowed bit vectors $\boldsymbol{b} \in \mathbb{B}_{n,m}$ and insertion locations, we fix the alignment between the positions in $\boldsymbol{x}$ and $\boldsymbol{y}$, which allows us to use the same bit vector for both $\boldsymbol{x}$ and $\boldsymbol{y}$, and sum over all the possible insertion locations, resulting in correlated samples.

[†] We have relabeled the time index of the reverse process as t to keep the notation simple.

2. We remove the summation over insertion locations and vocabulary items, and instead parameterize $\hat{R}_t^\theta(\boldsymbol{x}, \text{ins}(\boldsymbol{x}, i, w)) = \hat{\lambda}_t^\theta(\boldsymbol{x})\, \hat{p}_t^\theta(i, w \mid \boldsymbol{x})$ directly. This is an approximation only when $\boldsymbol{x}_0$ has contiguous sections of repeated. In all other cases, $\delta_{\text{ins}(\boldsymbol{x},i,w)}(\boldsymbol{y})$ will be 1 for only one insertion location.

Corollary 1. *A biased estimate of the upper bound on the negative log-likelihood in Proposition 2 for the parameterization in Equation (4) is given by*

$$\mathop{\mathbb{E}}_{t,m,\boldsymbol{b}} \left[\hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}]) - \gamma_t(n,m) \log \hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}]) \quad - \gamma_t(n,m) \sum_{\substack{i \in [\![m+1]\!], \\ w \in \mathbb{V}}} q(i, w \mid \boldsymbol{x}_0[\boldsymbol{b}]) \log \hat{p}_t^\theta(i, w \mid \boldsymbol{x}_0[\boldsymbol{b}]) \right]$$

where $\gamma_t(n,m) = \frac{\sigma_t\,(n-m)}{\bar{\sigma}_{0,t}\,\tilde{S}_{n,m,\sigma_t}}$ *with* $\tilde{S}_{n,m,\sigma_t} = [(1-\delta_0(m)) + \delta_0(m) S_{n,\sigma_t}]$, $S_{n,\sigma_t} = \sum_{k=0}^{\infty} \frac{n!\,\bar{\sigma}_{0,t}^k}{(n+k)!}$, $\boldsymbol{x}_0 = (n, \dot{\boldsymbol{x}}_0)$, *the expectation is over* $t \sim$ *Uniform*$[0,T]$, $(n-m) \sim$ Poisson$(\bar{\sigma}_{0,t})\,\delta\{m \geq 0\}$, $\boldsymbol{b} \sim$ *Uniform*$(\mathbb{B}_{n,m})$, *and* $q(i, w \mid \boldsymbol{x}_0[\boldsymbol{b}]) = \frac{\sum_{k \in s_i(\boldsymbol{b})} \delta_w(x_0^k)}{n-m}$ *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.*

The joint probability distribution over positions and tokens can be parameterized easily using a standard transformer encoder with time conditioning. Let $f^\theta : \mathbb{V}^n \times [0,1] \to \mathbb{R}^{n \times d}$ denote the transformer with bi-directional attention (without the final linear projection), which can take time $t \in [0,1]$ as an input in addition to the sequence $\boldsymbol{x}$. For each position $i \in [\![n]\!]$ the corresponding output of the transformer backbone $f^\theta(\boldsymbol{x}; t)^i \in \mathbb{R}^d$ is passed through the linear unembedding layer $U^\theta : \mathbb{R}^d \to \mathbb{R}^{|\mathbb{V}|}$ to get the probability of (i, w) pair in the sequence.

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

Parameterizing the rate of insertion $\hat{\lambda}_t^\theta$, non-negative real number is more challenging [Campbell et al., 2023, Patel et al., 2025]. We find that parameterizing the length-to-go directly using a dedicated input position in the transformer backbone works the best. Therefore, we parameterize $P\{N_0 - N_t = l \mid \boldsymbol{X}_t = \boldsymbol{x}_t\}$ and denote it as $\hat{p}_{\text{len}}^\theta(l \mid \boldsymbol{x}_t)$. It can be shown that the rate of insertion is given by

$$\hat{\lambda}_t^\theta(\boldsymbol{x}) = \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{l \sim p_{t,\,\text{len}}^\theta} \frac{l}{\tilde{S}_{l,m,\sigma_t}}$$

where $\tilde{S}_{l,m,\sigma_t}$ is as defined in Corollary 1, and the expectation is easy to compute in analytical form. The generation is simulated using the Euler method as described in Algorithm 2. We call the parameterization Diffusion-based Insertion Language Model with Single 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 in the Bernoulli probability $\hat{\lambda}_t^\theta(\boldsymbol{x})\Delta t$ in Algorithm 2 is replaced with a stopping probability $p_{\text{stop}}^\theta(\cdot \mid \boldsymbol{x})$. 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 $p_{\text{stop}}^\theta(\text{stop} \mid \boldsymbol{x}) < \zeta$. This means that there is no way to trade-off the quality of the generated sequence reducing Δt, or equivalently, increasing the number of sampling steps. Our method, like most other diffusion-based generative models, allows for this trade-off.

Algorithm 2 Generation using Euler method for DILM-S

Require: Sequence $\boldsymbol{x} = (m, (\boldsymbol{v}, \boldsymbol{u}))$, time $t \in [0, 1]$, steps maxsteps

$\Delta t \leftarrow \frac{1}{\text{maxsteps}}$
while $t < 1$ **do**
  $e \leftarrow \text{Uniform}[0, 1]$
  $\hat{\lambda}_t^\theta(\boldsymbol{x}) \leftarrow \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathbb{E}_{l \sim p^\theta_{t,\,\text{len}}(\boldsymbol{x})} \frac{l}{\tilde{S}_{l,m,\sigma_t}}$
  if $e \leq \hat{\lambda}_t^\theta(\boldsymbol{x})\, \Delta t$ **then**
    $i, w \sim \hat{p}_t^\theta(i, w \mid \boldsymbol{x})$
    $\boldsymbol{v}' \leftarrow \texttt{concat}(\boldsymbol{v}, w)$
    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 $\boldsymbol{x}$

Algorithm 3 Generation using τ-leaping for DILM-M

Require: Sequence $\boldsymbol{x} = (m, \dot{\boldsymbol{x}})$, time $t \in [0, 1]$, steps maxsteps

$\Delta t \leftarrow \frac{1}{\text{maxsteps}}$
while $t < 1$ **do**
  $\mathcal{I} \leftarrow \{0, 1, \ldots, |\dot{\boldsymbol{x}}|\}$ // gap indices
  $\text{ins} \leftarrow []$; $\boldsymbol{x}' \leftarrow \boldsymbol{x}$ // clone
  for $i \in \mathcal{I}$ **do**
    $e \sim \text{Uniform}[0, 1]$
    if $e \leq \hat{\lambda}_t^\theta(\boldsymbol{x}, i)\, \Delta t$ **then**
      $w \sim \hat{p}_t^\theta(w \mid \boldsymbol{x}, i)$
      $\text{ins} \leftarrow \text{ins} \cup \{(i, w)\}$
    end if
  end for
  Build gap-mask $\mathbf{r} \in \{0, 1\}^m$, $\mathbf{r}[i] = 1$ iff $(i, \cdot) \in \text{ins}$
  if $\text{ins} \neq []$ **then**
    $\texttt{idx\_old} \leftarrow \texttt{arange}(m) + \texttt{CumSum}(\texttt{RollRight}(\mathbf{r}, 1))$
    $\texttt{idx\_ins} \leftarrow \texttt{arange}(m) + \texttt{CumSum}(\mathbf{r})$
    $\boldsymbol{x}' \leftarrow \texttt{scatter}(\boldsymbol{x}', \texttt{idx\_old}, \boldsymbol{x})$
    $\boldsymbol{x}' \leftarrow \texttt{scatter}(\boldsymbol{x}', \texttt{idx\_ins}, \text{ins})$
  end if
  $t \leftarrow t + \Delta t$; $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
end while
return $\boldsymbol{x}$

4.3.2 Independent Insertion Probabilities

We can parameterize the rate matrix of the generative process using independent, per-dimension probabilities [Campbell et al., 2022] as

$$\hat{R}_t^\theta(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m+1} \sum_{w \in \mathbb{V}} \lambda_t^\theta(\boldsymbol{x}, i)\, \hat{p}_t^\theta(w \mid \boldsymbol{x}, i,)\, \delta_{\text{ins}(\boldsymbol{x}, i, w)}(\boldsymbol{y}). \tag{5}$$

Applying the same approximations as done for Corollary 1, we get the following biased estimate of the upper bound on the negative log-likelihood, which we can use to train the model.

Corollary 2. *A biased estimate of the upper bound on the negative log-likelihood in Proposition 2 for the parameterization in Equation (5) is given by*

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

where $\gamma_t(n, m) = \frac{\sigma_t\,(n-m)}{\bar{\sigma}_{0,t}\,\tilde{S}_{n,m,\sigma_t}}$ *with* $\tilde{S}_{n,m,\sigma_t} = [(1 - \delta_0(m)) + \delta_0(m) S_{n,\sigma_t}]$, $S_{n,\sigma_t} = \sum_{k=0}^{\infty} \frac{n!\, \bar{\sigma}_{0,t}^k}{(n+k)!}$, $\boldsymbol{x}_0 = (n, \dot{\boldsymbol{x}}_0)$, *the expectation is over* $t \sim \textit{Uniform}[0, T]$, $(n - m) \sim \text{Poisson}(\bar{\sigma}_{0,t})\, \delta\{m \geq 0\}$, $\boldsymbol{b} \sim \textit{Uniform}(\mathbb{B}_{n,m})$, *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.*

Algorithm 3 shows an efficient τ-leaping based sampling procedure for the parameterization in Equation (5), which can be used to insert multiple tokens per step. Additionally, this algorithm can be applied on a mini-batch of sequences in parallel.[‡] We call the parameterization MDILM-S, where M stands for multiple insertions.

[‡]We provide a simplified implementation of Algorithm 3 in PyTorch in Listing D.1.

5 Empirical Evaluation

We split our experiments in three 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 left-to-right autoregressive language models (ARMs), and masked diffusion models (MDMs).

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]. We also use the first token position to predict the length-to-go probabilities. 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.

Planning Task. The simple star graph task [Bachmann and Nagarajan, 2024] and its harder variable length variant [Patel et al., 2025] is a minimal task designed to exemplify limitations of ARMs and MDMs. [§] This is a sequence-to-sequence task where the source sequence contains randomly permuted edges of a star shaped graph, followed by a start and a 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. While MDMs, which can model all the tokens simultaneously, are able to generate the correct sequence, but 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.

Table 1: seq. denotes full sequence accuracy, and tok. denotes per token accuracy.

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 Language Modeling

For all the language modeling experiments, we use a transformer backbone with 12 layers, 768 hidden dimensions, and 12 attention heads with approximately 87 million non-embedding parameters and train it from scratch using AdamW optimizer on 4 A100 GPUs.

5.1.1 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 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 from each

[§]We use dataset from [Patel et al., 2025].

model computed as:

$$\text{NLL}(\boldsymbol{x}) = -\frac{1}{|\boldsymbol{x}|}\sum_{i=1}^{|\boldsymbol{x}|}\log p^{\text{LLM}}(x_i|\boldsymbol{x}_{1:i-1}).$$

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 our formulation is the ability to trade-off sample quality and number of forward passes. Additionally, we also observe improvements in the entropy which is a good indicator of the diversity of the generated samples. As shown in Figure 2, the generative perplexity of both DILM-S and DILM-M improves as we increase the number of sampling steps.

5.1.2 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 a batch size of 512, and train for 300 thousand steps. Table 2 reports the per-token negative log-likelihood for 1000 unconditionally generated sequences from the model. We can see that DILM-S and DILM-M perform better than ILM and MDM. However, on longer sequences, we observe token repetitions which shows up in the lower entropy values.

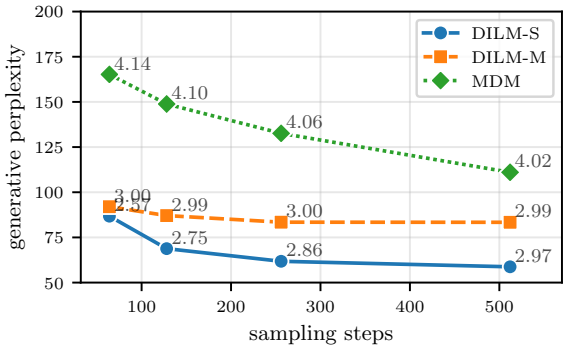

Figure 2: Generative perplexity and entropy vs. sampling steps on LM1B. The entropy values are annotated on the plots.

Table 2: Unconditional generation quality (per-token NLL under Llama 3.2 3B). Dataset rows show training data statistics.

	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

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 approximations, we derived a low-variance learning objective for both the parameterizations, and also demonstrate 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. Multi-token insertion model that can insert contiguous spans of tokens is promising direction for further exploration.

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*.

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

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.

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*.

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.

Appendix

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 $\{1, 2, \dots, n\}$
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 $P\{\boldsymbol{X}_s = \boldsymbol{x} \mid \boldsymbol{X}_t = \boldsymbol{x}'\}$
$\underline{p}_{k\|k'}(\boldsymbol{x} \mid \boldsymbol{x}')$	Transition probability $P\{\underline{\boldsymbol{X}}_k = \boldsymbol{x} \mid \underline{\boldsymbol{X}}_{k'} = \boldsymbol{x}'\}$
$\delta_{\mathbb{A}}(a)$	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.
$\lvert\cdot\rvert : \mathbb{B}_n \to \mathbb{N}$	The number of 1s in a bit vector; $\lvert\boldsymbol{b}\rvert = \sum_{i=1}^{n} b^i$
$\mathrm{Idx}(\boldsymbol{b}) \coloneqq \{i \in [\![n]\!] \mid b^i = 1\}$	The set of indices of the ones in $\boldsymbol{b}$
$\mathrm{del}(\boldsymbol{x}, i)$	$\mathrm{del}(x^1 \dots x^i \dots x^n, i) = x^1 \dots x^{i-1} x^{i+1} \dots x^n$
$\mathrm{ins}(\boldsymbol{x}, i, a)$	$\mathrm{ins}(x^1 \dots x^i \dots x^n, i, a) = x^1 \dots x^{i-1} a\, x^{i+1} \dots x^n$
$\mathrm{Range}_{\mathrm{ins}}(\boldsymbol{x})$	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.

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 $t \geq 0$,*

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

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

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

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

$$p(x, u + v;\, t) = p(x, u;\, t)\; p(x, v;\, t + u). \tag{6}$$

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 6, 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 3 shows an example of 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

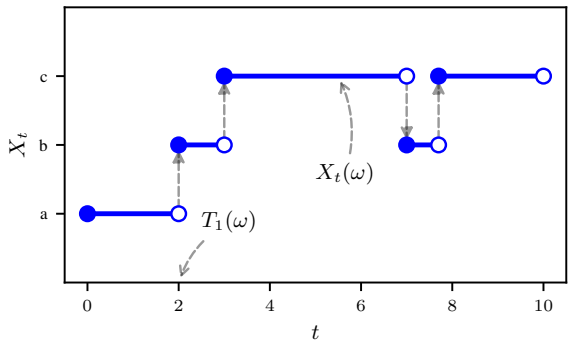

Figure 3: Sample path of a right-continuous CTMC with states $\mathbb{X} = \{a, b, c\}$.

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.

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*

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

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,

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

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

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

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

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

□

Fact 4. *Denote $P\{X_{t+u} = x' \mid X_t = x\}$ 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,

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

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

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

□

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

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

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 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 Noising Process

Lemma 1 (Deletion DTMC)**.** *Let $\underset{\sim}{\boldsymbol{X}}_k$ 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 $\underset{\sim}{\boldsymbol{X}}_k$ is a DTMC with the following one-step transition kernel and transition probability, respectively:*

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

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

Remark 1. 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

$$\begin{aligned}
&\sum_{\boldsymbol{x}\in\mathbb{X}_n}\sum_{\boldsymbol{b}\in\mathbb{B}_{l,n}}\sum_{\boldsymbol{c}\in\mathbb{B}_{n,n-1}} \delta_{\dot{\boldsymbol{z}}[\boldsymbol{b}]}(\dot{\boldsymbol{x}})\;\delta_{\dot{\boldsymbol{x}}[\boldsymbol{c}]}(\dot{\boldsymbol{y}}) \\
&\overset{(a)}{=} \sum_{\boldsymbol{b}\in\mathbb{B}_{l,n}}\sum_{\boldsymbol{c}\in\mathbb{B}_{n,n-1}} \delta_{\dot{\boldsymbol{z}}[\boldsymbol{b}][\boldsymbol{c}]}(\dot{\boldsymbol{y}}) \\
&\overset{(b)}{=} \sum_{\boldsymbol{b}\in\mathbb{B}_{l,n}}\sum_{\boldsymbol{a}\in\mathbb{A}_{l,n-1,\boldsymbol{b}}} \delta_{\dot{\boldsymbol{z}}[\boldsymbol{a}]}(\dot{\boldsymbol{y}}) \\
&= \sum_{\boldsymbol{a}\in\mathbb{B}_{l,n-1}} (l-n+1)\delta_{\dot{\boldsymbol{z}}[\boldsymbol{a}]}(\dot{\boldsymbol{y}})
\end{aligned} \tag{7}$$

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 have to become 1 to obtain a valid $\boldsymbol{b}$.

Proof. The transition kernel is given by

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

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 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 done in the Remark 1, 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.

$$\begin{aligned}
&= \sum_{\boldsymbol{b}_1 \in \mathbb{B}_{n_0, n_0 - 1}} \frac{1}{n_0} \sum_{\boldsymbol{b}_2 \in \mathbb{B}_{n_0 - 1, \, n_0 - 2}} \frac{1}{n_0 - 1} \delta_{(\dot{\boldsymbol{z}}_0[\boldsymbol{b}_1])[\boldsymbol{b}_2]}(\dot{\boldsymbol{z}}_2) \\
&= \sum_{\boldsymbol{b} \in \mathbb{B}_{n, n-2}} \frac{1 \cdot 2}{n \, (n-1)} \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{z}}_2) \qquad (8)
\end{aligned}$$

In the final step, we removed the summation over $\boldsymbol{b}_1$ using the same argument as we used in Equation (7). 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.

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

□

Proposition 1 (Transition Probability)**.** *Let $\boldsymbol{X}_t$ be a continuous time stochastic process described in equation 2, with transition rate $\lambda_t((r, \dot{\boldsymbol{x}})) = \sigma_t \delta\{r > 0\}$ 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 $s \leq t$ given by*

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

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 $\underline{\boldsymbol{X}}_k = (\underline{N}_k, \dot{\underline{\boldsymbol{X}}}_k)$, respectively, where N_t and $\underline{N}_k$ are the lengths of the sequences at time t and k, respectively. Using this, we can write the transition probability as

$$\begin{aligned}
&P\{\boldsymbol{X}_t = (m, \dot{\boldsymbol{y}}) \mid \boldsymbol{X}_s = (n, \dot{\boldsymbol{x}})\} \\
&= P\{N_t = m, \dot{\boldsymbol{X}}_t = \dot{\boldsymbol{y}} \mid N_s = n, \dot{\boldsymbol{X}}_s = \dot{\boldsymbol{x}}\} \\
&= P\{\dot{\boldsymbol{X}}_t = \dot{\boldsymbol{y}} \mid \dot{\boldsymbol{X}}_s = \dot{\boldsymbol{x}}, N_s = n, N_t = m\} P\{N_t = m \mid N_s = n, \dot{\boldsymbol{X}}_s = \dot{\boldsymbol{x}}\} \\
&\overset{(a)}{=} P\{\dot{\underline{\boldsymbol{X}}}_{n-m} = \dot{\boldsymbol{y}} \mid \dot{\underline{\boldsymbol{X}}}_0 = \dot{\boldsymbol{x}}, \underline{N}_0 = n\} P\{N_t = m \mid N_s = n\} \\
&\overset{(b)}{=} P\{\underline{\boldsymbol{X}}_{n-m} = (m, \dot{\boldsymbol{y}}) \mid \underline{\boldsymbol{X}}_0 = (n, \dot{\boldsymbol{x}})\} P\{N_t = m \mid N_s = n\}
\end{aligned} \tag{9}$$

Here, in (a), $P\{\dot{\boldsymbol{X}}_t = \dot{\boldsymbol{y}} \mid \dot{\boldsymbol{X}}_s = \dot{\boldsymbol{x}}, N_s = n, N_t = m\} = P\{\dot{\underline{\boldsymbol{X}}}_{n-m} = \dot{\boldsymbol{y}} \mid \dot{\underline{\boldsymbol{X}}}_0 = \dot{\boldsymbol{x}}, \underline{N}_0 = n\}$ 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 the deletions arrive with rate $\lambda_t((r, \dot{\boldsymbol{x}})) = \sigma_t \delta\{r > 0\}$ and therefore the number of deletions, up till there is nothing do delete, have Poisson distribution with rate $\bar{\sigma}_{s,t} = \int_s^t \sigma(u)\, du$. The general expression for $P\{N_t = m \mid N_s = n\}$ can be obtained by creating a special case for the absorbing state $m = 0$ and noting that the total probability is 1:

$$P\{N_t = m \mid N_s = n\} = \begin{cases} \frac{e^{-\bar{\sigma}_{s,t}} (\bar{\sigma}_{s,t})^{n-m}}{(n-m)!} & \text{if } 0 < m \leq n \\ 1 - \sum_{k=1}^{n} \frac{e^{-\bar{\sigma}_{s,t}} (\bar{\sigma}_{s,t})^{n-k}}{(n-k)!} & \text{if } m = 0. \end{cases} \tag{10}$$

We get the desired expression for the transition probability by of the CTMC by substituting the expression for $P\{\underline{\boldsymbol{X}}_{n-m} = (m, \dot{\boldsymbol{y}}) \mid \underline{\boldsymbol{X}}_0 = (n, \dot{\boldsymbol{x}})\}$ from Lemma 1, and for $P\{N_t = m \mid N_s = n\}$ from Equation (10), in Equation (9). □

C.2 Generative Process

Proposition 2. *The log-likelihood of the data* $\log p_{\text{data}}(x_0)$ *under the parameterized time reversal of the noising process which is a CTMC on a finite state space is bounded from below by*

$$\mathbb{E} \sum_{y \neq x} \left[-\hat{R}_t^\theta(x, y) + \frac{p_{t|0}(y|x_0)}{p_{t|0}(x|x_0)} R_t(y, x) \log \hat{R}_t^\theta(x, y) \right] + C, \tag{3}$$

where the expectation is over $t \sim \textit{Uniform}[0, 1],\ x \sim p_{t|0}(x \mid x_0)$*, and* C *is independent of* θ*.*

Proof. The result can be proved for any regular CTMC using Dynkin's formula [Hanson, 2007] for the change of measure as discussed in Lou et al. [2024], Campbell et al. [2022].

In order to provide useful intuition for the result, we instead use elementary techniques to provide an informal result 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:

$$\begin{aligned}
\log(1 + x) &= x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots \\
&= x - \frac{x^2}{2} + o(x^2)
\end{aligned} \tag{11}$$

2. For $a, b > 0$,

$$\begin{aligned}\log(ab + o(a)) &= \log(a(b + o(1))) \\ &= \log(a) + \log(b + o(1)) \\ &= \log(a) + \log(b(1 + o(1))) \\ &= \log(a) + \log(b) + \log(1 + o(1)) \\ &= \log(a) + \log(b) + o(1)\end{aligned} \tag{12}$$

3. Bayes rule and the Markov property implies

$$\begin{aligned}q(x_k \mid x_{k+1}, x_0) &= \frac{q(x_k \mid x_0)}{q(x_{k+1} \mid x_0)} q(x_{k+1} \mid x_k) \\ &= r(k, k+1) q(x_{k+1} \mid x_k),\end{aligned} \tag{13}$$

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 being, 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 with 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. Following are the expressions for the conditional probabilities in terms of the forward and reverse rate matrices:

$$\begin{aligned}q(x_k \mid x_{k+1}, x_0) &= r(k, k+1)\left(\delta_{x_k}(x_{k+1}) + R_k(x_k, x_{k+1})\Delta t + o(\Delta t)\right) \\ p(x_k \mid x_{k+1}) &= \delta_{x_k}(x_{k+1}) + \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)\end{aligned} \tag{14}$$

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

$$\begin{aligned}
\log(\delta + \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)) &= \delta \log(1 + \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)) \\
&\quad + \bar{\delta} \log(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)) \\
&= \delta(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)) \\
&\quad + \bar{\delta} \log(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t) + o(\Delta t) \quad \text{(by 11)} \\
&= \delta(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)) \\
&\quad + \bar{\delta} \log(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t) + o(\Delta t) \\
&= \delta(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t) \\
&\quad + \bar{\delta}(\log(\hat{R}_{k+1}(x_{k+1}, x_k)) + \log(\Delta t) + o(1)) + o(\Delta t) \quad \text{(by 12)}
\end{aligned} \tag{15}$$

Therefore,

$$\begin{aligned}
&- \text{CE}[q(x_k \mid x_{k+1}, x_0) \,\|\, p(x_k \mid x_{k+1})] \\
&= \sum_{x_k} r \left\{ \delta \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + \bar{\delta}\Delta t R_k(x_k, x_{k+1}) \log(\hat{R}_{k+1}(x_{k+1}, x_k)) + \Delta t \log(\Delta t) + o(\Delta t) \right\} \\
&= \sum_{x_k} r \left\{ \delta \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + \bar{\delta}\Delta t R_k(x_k, x_{k+1}) \log(\hat{R}_{k+1}(x_{k+1}, x_k)) + o(\Delta t) \right\} \\
&= \Delta t \hat{R}_{k+1}(x_{k+1}, x_{k+1}) + \Delta t \sum_{x_k \neq x_{k+1}} r R_k(x_k, x_{k+1}) \log(\hat{R}_{k+1}(x_{k+1}, x_k)) + o(\Delta t)
\end{aligned} \tag{16}$$

Similarly,

$$\begin{aligned}
-H[q(x_k \mid x_{k+1}, x_0)] &= \sum_{x_k} r \{\delta + R_k(x_k, x_{k+1})\Delta t + o(\Delta t)\} \log r \{\delta + R_k(x_k, x_{k+1})\Delta t + o(\Delta t)\} \\
&= \sum_{x_k} r \left\{ \delta R_k(x_k, x_{k+1})\Delta t + \bar{\delta}\Delta t R_k(x_k, x_{k+1}) \log(R_k(x_k, x_{k+1})) + o(\Delta t) \right\} \\
&\quad + \sum_{x_k} \delta r \log r + r\Delta t R_k(x_k, x_{k+1}) \log r \\
&= \Delta t R_k(x_{k+1}, x_{k+1}) + \Delta t \sum_{x_k \neq x_{k+1}} r R_k(x_k, x_{k+1}) \log(R_k(x_k, x_{k+1})) + o(\Delta t) \\
&\quad + \Delta t \sum_{x_k \neq x_{k+1}} R_k(x_k, x_{k+1}) r \log r \quad (\text{since } r \log r = 0 \text{ when } x_k = x_{k+1})
\end{aligned} \tag{17}$$

Putting the two together, we get

$$\begin{aligned}
L_k(x_0) = \mathop{\mathbb{E}}_{x_{k+1}} \sum_{x_k} r\Delta t \Bigg\{ & \underbrace{\left[\delta R_k(x_k, x_{k+1}) + \bar{\delta} R_k(x_k, x_{k+1}) \log(R_k(x_k, x_{k+1})) + \bar{\delta} R_k(x_k, x_{k+1}) \log r\right]}_{H \text{ term}} \\
& - \underbrace{\left[\delta \hat{R}_{k+1}(x_{k+1}, x_k) + \bar{\delta} R_k(x_k, x_{k+1}) \log(\hat{R}_{k+1}(x_{k+1}, x_k))\right]}_{\text{CE term}} + \frac{o(\Delta t)}{\Delta t} \Bigg\}
\end{aligned} \tag{18}$$

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 $\hat{R}$.

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

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

$$\sum_y \frac{q(y|x_0)}{q(x_t|x_0)} \delta_{x_t}(y) \hat{R}_t(x_t, y) = \hat{R}_t(x_t, x_t) = -\sum_{y \neq x_t} \hat{R}_t(x_t, y). \tag{19}$$

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 (19), 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. *A biased estimate of the upper bound on the negative log-likelihood in Proposition 2 for the parameterization in Equation (4) is given by*

$$\mathop{\mathbb{E}}_{t,m,\boldsymbol{b}} \left[\hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}]) - \gamma_t(n,m) \log \hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}]) \quad - \gamma_t(n,m) \sum_{\substack{i \in [\![m+1]\!], \\ w \in \mathbb{V}}} q(i, w \mid \boldsymbol{x}_0[\boldsymbol{b}]) \log \hat{p}_t^\theta(i, w \mid \boldsymbol{x}_0[\boldsymbol{b}]) \right]$$

where $\gamma_t(n,m) = \frac{\sigma_t\,(n-m)}{\bar{\sigma}_{0,t}\,\tilde{S}_{n,m,\sigma_t}}$ *with* $\tilde{S}_{n,m,\sigma_t} = [(1 - \delta_0(m)) + \delta_0(m) S_{n,\sigma_t}]$, $S_{n,\sigma_t} = \sum_{k=0}^{\infty} \frac{n!\,\bar{\sigma}_{0,t}^k}{(n+k)!}$, $\boldsymbol{x}_0 = (n, \dot{\boldsymbol{x}}_0)$, *the expectation is over* $t \sim$ *Uniform*$[0,T]$, $(n-m) \sim$ Poisson$(\bar{\sigma}_{0,t})\,\delta\{m \geq 0\}$, $\boldsymbol{b} \sim$ *Uniform*$(\mathbb{B}_{n,m})$, *and* $q(i, w \mid \boldsymbol{x}_0[\boldsymbol{b}]) = \frac{\sum_{k \in s_i(\boldsymbol{b})} \delta_w(x_0^k)}{n-m}$ *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 equation 3 with there 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

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

For $t>s$, and $m>0$, from equation 10 we have

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

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

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

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

$$\frac{P\{N_t=m+1\mid N_s=n\}}{P\{N_t=m\mid N_s=n\}}=\frac{n-m}{\bar{\sigma}_{s,t}}\frac{1}{\tilde{S}_{n,m,\sigma_t}}\tag{20}$$

For $r = m + 1$, we have

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

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

$$\begin{aligned} &- \hat{\lambda}_t(\boldsymbol{x}) + \sigma_t \frac{(n-m)}{\bar{\sigma}_{0,t}\, \tilde{S}_{n,m,\sigma_t}} \log \hat{\lambda}_t(\boldsymbol{x}) \frac{\sum_{\boldsymbol{y} \neq \boldsymbol{x}} K(\boldsymbol{y}, \boldsymbol{x}) P\{\underline{\boldsymbol{X}}_{n-m-1} = \dot{\boldsymbol{y}} \mid \underline{\boldsymbol{X}}_0 = \dot{\boldsymbol{x}}_0\}}{\sum_{\dot{\boldsymbol{z}} \in \mathbb{V}^{m+1}} P\{\underline{\boldsymbol{X}}_{n-m-1} = \dot{\boldsymbol{z}} \mid \underline{\boldsymbol{X}}_0 = \dot{\boldsymbol{x}}_0\} K(\boldsymbol{z}, \boldsymbol{x})} \\ &= -\hat{\lambda}_t(\boldsymbol{x}) + \sigma_t \frac{(n-m)}{\bar{\sigma}_{0,t}\, \tilde{S}_{n,m,\sigma_t}} \log \hat{\lambda}_t(\boldsymbol{x}) \end{aligned} \tag{21}$$

The token term (B):

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

$$\begin{aligned} &\mathop{\mathbb{E}}_{\boldsymbol{x} \sim p_{t|0}(\cdot|\boldsymbol{x}_0)} \sum_{\boldsymbol{y} \neq \boldsymbol{x}} \frac{P\{\underline{N}_t = r \mid \underline{N}_0 = n\}}{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_{\boldsymbol{x}_0[\boldsymbol{b}']}(\dot{\boldsymbol{y}})}{\binom{n}{m+1}} \sum_{i=1}^{m+1} \frac{\delta_{\mathsf{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_{\boldsymbol{x}_0[\boldsymbol{b}']}(\dot{\boldsymbol{z}})}{\binom{n}{m+1}} \sum_{i=1}^{m+1} \frac{\delta_{\mathsf{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_{\mathsf{del}(\boldsymbol{x}_0[\boldsymbol{b}'], i)}(\boldsymbol{x}_0[\boldsymbol{b}])}{\sum_{\boldsymbol{b}' \in \mathbb{B}_{n,m+1}} \sum_{i=1}^{m+1} \delta_{\mathsf{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) \end{aligned}$$

Now let's look at the numerator:

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

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. 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 $\pi(\boldsymbol{a}, i, k \mid \boldsymbol{x}_0, \boldsymbol{b})$ over the set $\{(\boldsymbol{a}, i, k) \in \mathbb{B}_{n,m}(\boldsymbol{x}_0, \boldsymbol{b}) \times [\![m+1]\!] \times s_i(\boldsymbol{a}) \mid \delta_w(\boldsymbol{x}_0^k) = 1\}$.

$$
\frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m,\boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \mathop{\mathbb{E}}_{\boldsymbol{a},i,k \sim \pi(\cdot \mid \boldsymbol{x}_0, \boldsymbol{b})} \log\left(\hat{p}^{\theta}_{\text{ins}}(i, x_0^k \mid \boldsymbol{x}_0[\boldsymbol{b}])\right).
$$

We get an unbiased estimate by taking a single sample $(\boldsymbol{a}, i, k) \sim \pi(\cdot \mid \boldsymbol{x}_0, \boldsymbol{b})$. We can get a *biased* estimate by taking $(n-m)$ correlated samples by fixing $\boldsymbol{a}$ to be the same for all samples and equal to $\boldsymbol{b}$:

$$
\begin{aligned}
&\frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m,\boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \frac{1}{n-m} \sum_{i=1}^{m+1} \sum_{k \in s_i(\boldsymbol{b})} \sum_{w \in \mathbb{V}} \delta_w(\boldsymbol{x}_0^k) \log\left(\hat{p}^{\theta}_{\text{ins}}(i, x_0^k \mid \boldsymbol{x}_0[\boldsymbol{b}])\right) \\
&= \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m,\boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \sum_{\substack{i \in [\![m+1]\!], \\ w \in \mathbb{V}}} p^{\text{target}}_{\text{ins}}(i, w \mid \boldsymbol{x}_0, \boldsymbol{b}) \log\left(\hat{p}^{\theta}_{\text{ins}}(i, w \mid \boldsymbol{x}_0[\boldsymbol{b}])\right). \qquad (22)
\end{aligned}
$$

where $p^{\text{target}}_{\text{ins}}(i, w \mid \boldsymbol{x}_0, \boldsymbol{b}) = \frac{1}{n-m} \sum_{k \in s_i(\boldsymbol{b})} \delta_w(\boldsymbol{x}_0^k)$ 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. (21) and Eq. (22) together we get the final expression. □

Corollary 3. *A biased estimate of the upper bound on the negative log-likelihood in Proposition 2 for the parameterization in Equation (5) is given by*

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

where $\gamma_t(n,m) = \frac{\sigma_t \, (n-m)}{\bar{\sigma}_{0,t} \, \tilde{S}_{n,m,\sigma_t}}$ *with* $\tilde{S}_{n,m,\sigma_t} = [(1 - \delta_0(m)) + \delta_0(m) S_{n,\sigma_t}]$, $S_{n,\sigma_t} = \sum_{k=0}^{\infty} \frac{n! \, \bar{\sigma}_{0,t}^k}{(n+k)!}$, $\boldsymbol{x}_0 = (n, \dot{\boldsymbol{x}}_0)$, *the expectation is over* $t \sim$ *Uniform*$[0, T]$, $(n-m) \sim$ Poisson$(\bar{\sigma}_{0,t}) \, \delta\{m \geq 0\}$, $\boldsymbol{b} \sim$ *Uniform*$(\mathbb{B}_{n,m})$, *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 $\boldsymbol{x} = (m, \dot{\boldsymbol{x}})$, let

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

Replacing the forward rate matrix in equation 3 with its components, we get

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim p_{t|0}(\cdot|\boldsymbol{x}_0)} \sum_{\boldsymbol{y} \neq \boldsymbol{x}} -\hat{R}_t(\boldsymbol{x}, \boldsymbol{y}) \qquad \underbrace{+ \sum_{\boldsymbol{y} \neq \boldsymbol{x}} K_t(\boldsymbol{y}, \boldsymbol{x}) \frac{p_{t|0}(\boldsymbol{y} \mid \boldsymbol{x}_0)}{p_{t|0}(\boldsymbol{x} \mid \boldsymbol{x}_0)} \lambda_t(\boldsymbol{y}) \log\left(\hat{R}_t(\boldsymbol{x}, \boldsymbol{y})\right)}_{\text{the token term}(B)}$$

For term B, we proceed as in the proof C.2 to get

$$= \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m, \boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \frac{\sum_{\boldsymbol{b}' \in \mathbb{B}_{n,m+1}} \sum_{i=1}^{m+1} \delta_{\text{del}(\boldsymbol{x}_0[\boldsymbol{b}'], i)}(\boldsymbol{x}_0[\boldsymbol{b}])}{\sum_{\boldsymbol{b}' \in \mathbb{B}_{n,m+1}} \sum_{i=1}^{m+1} \delta_{\text{del}(\boldsymbol{x}_0[\boldsymbol{b}'], i)}(\boldsymbol{x}_0[\boldsymbol{b}])} \log\left(\hat{R}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{x}_0[\boldsymbol{b}'])\right)$$

Proceeding as in the proof C.2 with the same correlated sampling assumption, and then inserting the expression for $\hat{R}_t$, we get

$$\frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m, \boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \frac{1}{n-m} \sum_{i=1}^{m+1} \sum_{k \in s_i(\boldsymbol{b})} \sum_{w \in \mathbb{V}} \delta_w(\boldsymbol{x}_0^k) \log\left(\hat{R}_t(\boldsymbol{x}_0[\boldsymbol{b}], \text{ins}(\boldsymbol{x}_0[\boldsymbol{b}], i, w))\right)$$

$$\frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m, \boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \frac{1}{n-m} \sum_{i=1}^{m+1} \sum_{k \in s_i(\boldsymbol{b})} \sum_{w \in \mathbb{V}} \delta_w(\boldsymbol{x}_0^k) \log\left(\hat{\lambda}_t(\boldsymbol{x}_0[\boldsymbol{b}], i)\, \hat{p}_{\text{ins}}(w \mid \boldsymbol{x}_0[\boldsymbol{b}], i; t)\right)$$

$$= \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m, \boldsymbol{b}} \frac{1}{\tilde{S}_{n,m,\sigma_t}} \sum_{i=1}^{m+1} \left[\sum_{k \in s_i(\boldsymbol{b})} \sum_{w \in \mathbb{V}} \delta_w(\boldsymbol{x}_0^k) \log\left(\hat{\lambda}_t(\boldsymbol{x}_0[\boldsymbol{b}], i)\right) + \sum_{k \in s_i(\boldsymbol{b})} \log\left(\hat{p}_{\text{ins}}(x_0^k \mid \boldsymbol{x}_0[\boldsymbol{b}], i; t)\right) \right]$$

$$= \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m, \boldsymbol{b}} \frac{1}{\tilde{S}_{n,m,\sigma_t}} \sum_{i=1}^{m+1} \left[|s_i(\boldsymbol{b})| \log\left(\hat{\lambda}_t(\boldsymbol{x}_0[\boldsymbol{b}], i)\right) + \sum_{k \in s_i(\boldsymbol{b})} \log\left(\hat{p}_{\text{ins}}(x_0^k \mid \boldsymbol{x}_0[\boldsymbol{b}], i; t)\right) \right]$$

□

D Implementation Details

D.1 Computing $\frac{P\{N_t = m-1 | N_s = n\}}{P\{N_t = m | N_s = n\}}$

For $t > s$, and $m > 0$, from equation 10 we have

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

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

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

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 the 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 $\frac{P\{N_t=m+1|N_s=n\}}{P\{N_t=m|N_s=n\}}$

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 hypergeometric confluent 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 (e.g. 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)  # shape (batch, 1)
    x = x.unsqueeze(1)  # shape (batch, 1)
    # n = n.to(torch.float64)
    # x = x.to(torch.float64)

    # 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,)
```

Listing: Multi-token insertion step in PyTorch

```
def multi_token_step(
    x_t: torch.Tensor,  # shape (batch, seq_len)
    model: torch.nn.Module,
    sigma_t: torch.Tensor,  # shape (batch,) - noise rate
    bar_sigma_t: torch.Tensor,  # shape (batch,) - total noise
    dt: float = 0.1,
    max_length: int = 512,
    pad_token_id: int = 0,
) -> torch.Tensor:
    batch_size, seq_len = x_t.shape
    device = x_t.device

    # Create attention mask (1 for non-pad tokens)
    attention_mask = (x_t != pad_token_id)

    # Create positions (incremental positions for non-pad tokens)
    positions = (attention_mask.cumsum(dim=-1) - 1).clamp(min=0)

    # Get model predictions
    logits, length_logits = model(
        x_t,
        bar_sigma_t.unsqueeze(-1).expand(-1, seq_len),  # broadcast to (batch, seq_len)
        attention_mask,
        positions=positions
    )

    # Get mean delta length prediction
    delta_l = torch.arange(0, max_length, device=device)[None, None, :]  # (1, 1, max_length)
    p_length = torch.softmax(length_logits, dim=-1)  # (batch, seq_len, max_length)
    mean_delta_l = (p_length * delta_l).sum(dim=-1)  # (batch, seq_len)

    # Calculate insertion probability
    p = (
        sigma_t.unsqueeze(-1) * mean_delta_l / (bar_sigma_t.unsqueeze(-1) + 1e-6)
    ) * dt

    # Only predict at non-pad positions
    p = torch.where(attention_mask, p, torch.zeros_like(p))

    # Don't predict if it would exceed max length
    current_length = attention_mask.sum(-1)
```

```
    p = p * (current_length < max_length).unsqueeze(-1).float()

    # Sample whether to predict at each position
    predict = torch.rand_like(p) < p  # shape (batch, seq_len)

    # If no predictions, return unchanged sequence
    if not predict.any().item():
        return x_t

    # Prepare logits for sampling (set non-predicted positions to pad)
    logits_for_sampling = logits.clone()
    logits_for_sampling[~predict] = -torch.inf
    logits_for_sampling[..., pad_token_id].masked_fill(~predict, 100.0)

    # Sample new tokens using Gumbel-Max trick
    pred_vocab_index = sample(logits_for_sampling)

    # Insert tokens into sequence
    x_s = torch.full_like(x_t, pad_token_id)

    # Calculate new positions for existing and inserted tokens
    inc_positions = torch.arange(seq_len, device=device).unsqueeze(0).expand(batch_size, -1)

    # Positions for existing tokens (shifted by cumulative insertions before them)
    existing_tokens_new_positions = (
        inc_positions + predict.roll(shifts=1, dims=-1).int().cumsum(dim=-1)
    ).clamp(max=seq_len - 1)

    # Positions for newly inserted tokens
    inserted_tokens_new_positions = (
        inc_positions + predict.int().cumsum(dim=-1)
    ).clamp(max=seq_len - 1)

    # Batch indices for scatter operations
    batch_index = torch.arange(batch_size, device=device).unsqueeze(-1).expand(-1, seq_len)

    # Place existing tokens in their new positions
    x_s.scatter_(dim=-1, index=existing_tokens_new_positions, src=x_t)

    # Place newly predicted tokens
    pred_flat = pred_vocab_index[predict]
    x_s[batch_index[predict], inserted_tokens_new_positions[predict]] = pred_flat

    return x_s
```

E Unconditional Generation Examples

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.