Edit Flows: Flow Matching with Edit Operations

arXiv 2025

Marton Havasi[1], Brian Karrer[1], Itai Gat[1], Ricky T. Q. Chen[1]

[1]FAIR at Meta

2025. 07. 04

Seungwoo Yoo

KAIST

Motivation

Non-autoregressive models have become the standard across high-dimensional modalities, including images and videos.

Esser *et al.*, Stable Diffusion 3, ICML 2024

Wan 2.1, Team Wan, arXiv 2025

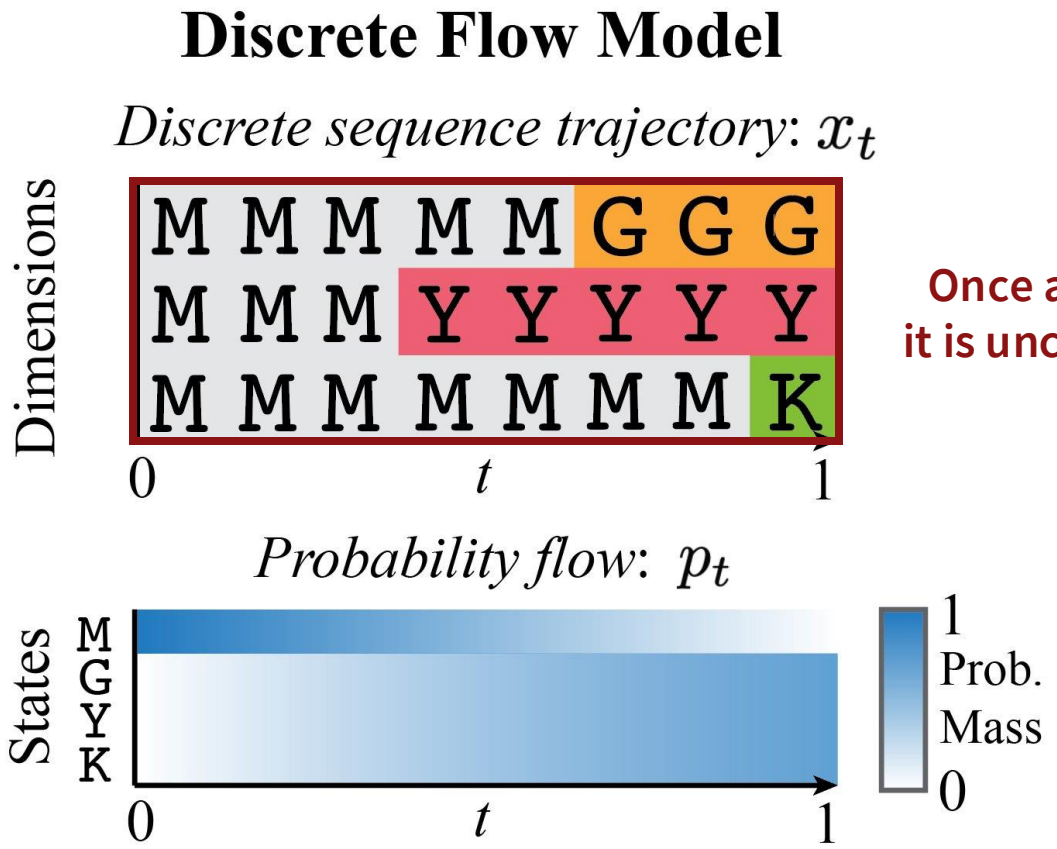
Motivation

Recent efforts have aimed to extend these advances to discrete code and text generation, as exemplified by LLaDA [Nie *et al.*, 2025] and Mercury [Inception Labs, 2025].

Mercury, Inception Labs

Motivation

However, these approaches do not support token insertion or deletion—operations that are fundamental to sequence generation and editing.



Once a token is unmasked, it is unchanged until the end!

Generative Flows on Discrete State-Spaces, Campbell *et al.*, ICML 2024

Motivation

This work presents **Edit Flows**, an extension of Discrete Flow Matching [Gat *et al.*, 2024] that enables the training of discrete flow models generating variable-length sequences.

Insert-only

```
def is_prime(n: int) -> bool:
```

Insert + Delete + Substitute

```
def is_prime(n: int) -> bool:
    if n <= 1:
        return True
    for i in range(2, n):
        if i % n == 0:
            return True
    return False
```

Continuous-time Markov Chains (CTMCs)

Consider **a continuous-time Markov chain (CTMC)** over a **discrete state space** $\mathcal{X}$, a Markov process that generates trajectories $(X_t)_{t\in[0,1]}$ characterized by:

$$\mathbb{P}(X_{t+h} = x | X_t = x_t) = \delta_{x_t}(x) + hu_t(x|x_t) + o(h)$$

where u_t denotes a **rate**-the infinitesimal transition probabilities between states.

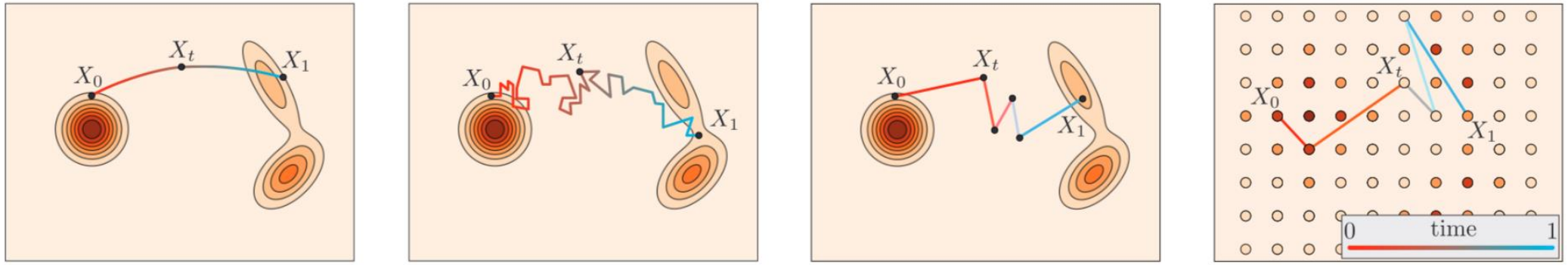
Continuous-time Markov Chains (CTMCs)

Consider **a continuous-time Markov chain (CTMC)** over a **discrete state space** $\mathcal{X}$, a Markov process that generates trajectories $(X_t)_{t\in[0,1]}$ characterized by:

$$\mathbb{P}(X_{t+h} = x | X_t = x_t) = \delta_{x_t}(x) + hu_t(x|x_t) + o(h)$$

where u_t denotes a **rate**-the infinitesimal transition probabilities between states.

Sampling from a CTMC is done by iteratively applying the above formula, which often starts from an initial sample $x_0 \sim p(x_0)$ and ends at $x_1 \sim q(x_1)$.



(a) Flow **(b)** Diffusion **(c)** Jump **(d)** CTMC

Lipman *et al.*, Flow Matching Guide and Code, NeurIPS 2024 Tutorial

Continuous-time Markov Chains (CTMCs)

The rate $u_t(x|x_t)$ is the infinitesimal probabilities of transitioning from a state x_t to any other state x at time t, which must satisfy the **rate condition**

$$u_t(x|x_t) \geq 0 \text{ for all } x \neq x_t, \sum_x u_t(x|x_t) = 0$$

for the probability mass in the previous slide to be valid (*i.e.*, sum to one).

Note that this implies that

$$u_t(x_t|x_t) = -\sum_{x \neq x_t} u_t(x|x_t)$$

Continuous-time Markov Chains (CTMCs)

In CTMC, state transitions occur as discrete jumps over the course of continuous time:

$$\mathbb{P}(X_{t+h} = x | X_t = x_t) = \delta_{x_t}(x) + hu_t(x|x_t) + o(h)$$

Example adapted from a talk by Andrew Campbell and Jason Yim.

Continuous-time Markov Chains (CTMCs)

In CTMC, state transitions occur as discrete jumps over the course of continuous time:

$$\mathbb{P}(X_{t+h} = x | X_t = x_t) = \delta_{x_t}(x) + hu_t(x|x_t) + o(h)$$

Example adapted from a talk by Andrew Campbell and Jason Yim.

Continuous-time Markov Chains (CTMCs)

In CTMC, state transitions occur as discrete jumps over the course of continuous time:

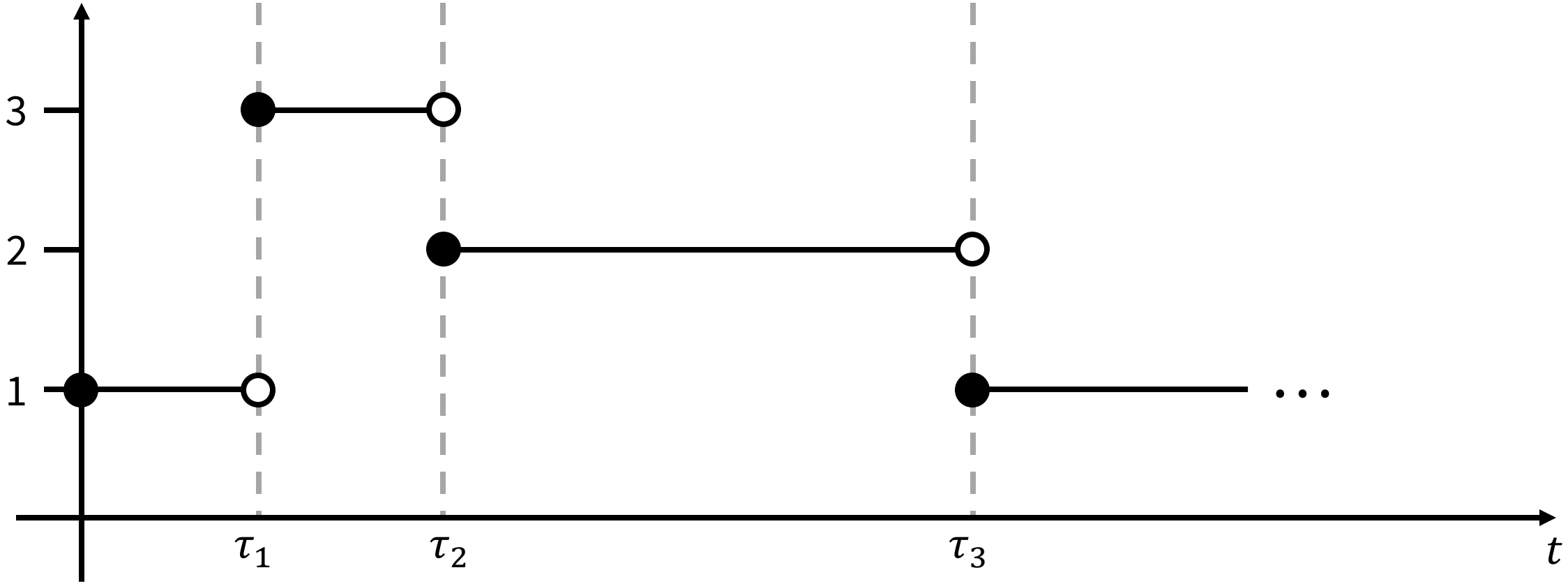
$$\mathbb{P}(X_{t+h} = x | X_t = x_t) = \delta_{x_t}(x) + hu_t(x|x_t) + o(h)$$

Example adapted from a talk by Andrew Campbell and Jason Yim.

Continuous-time Markov Chains (CTMCs)

In CTMC, state transitions occur as discrete jumps over the course of continuous time:

$$\mathbb{P}(X_{t+h} = x | X_t = x_t) = \delta_{x_t}(x) + hu_t(x|x_t) + o(h)$$



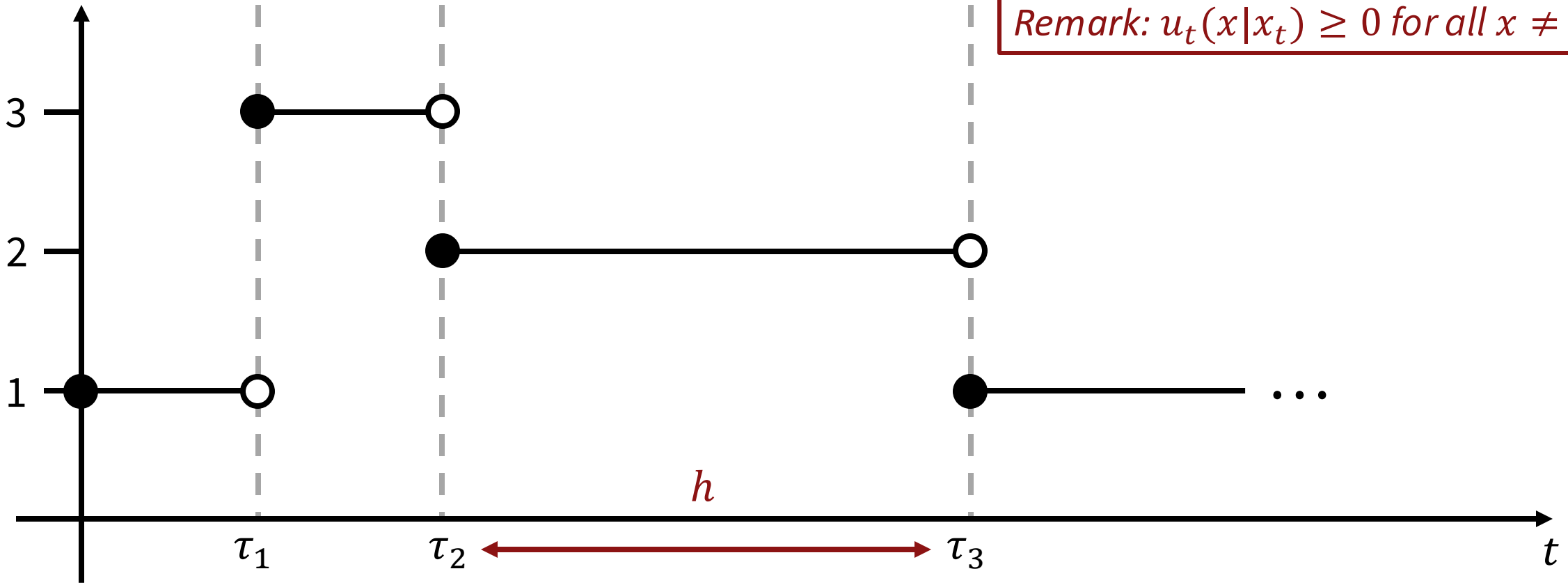
Example adapted from a talk by Andrew Campbell and Jason Yim.

Continuous-time Markov Chains (CTMCs)

In CTMC, state transitions occur as discrete jumps over the course of continuous time:

$$\mathbb{P}(X_{t+h} = x | X_t = x_t) = \delta_{x_t}(x) + h u_t(x|x_t) + o(h)$$

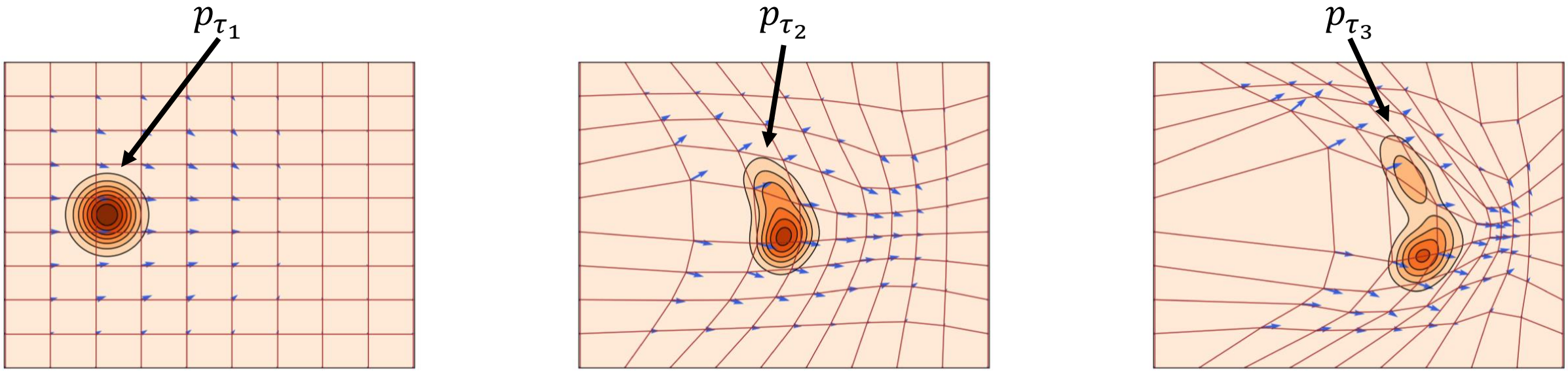
Remark: $u_t(x|x_t) \geq 0$ *for all* $x \neq x_t$



Example adapted from a talk by Andrew Campbell and Jason Yim.

Continuous-time Markov Chains (CTMCs)

A rate u_t is said to *generate* a **probability path** p_t if the time marginals of the associated CTMC are samples from p_t, *i.e.*, $X_t \sim p_t$.



Remark: The probability path is shown in continuous state space for visualization purposes only.

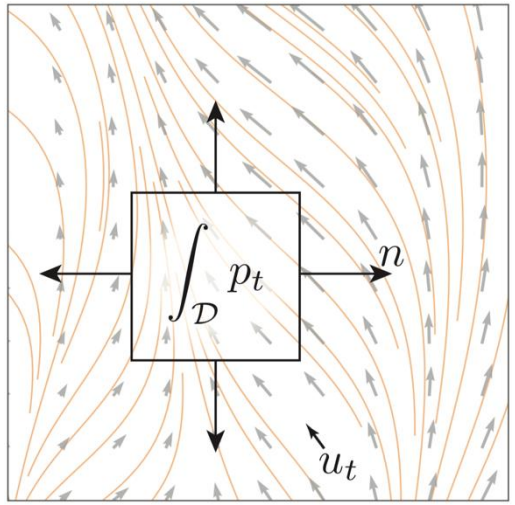
Lipman *et al.*, Flow Matching Guide and Code, NeurIPS 2024 Tutorial

Continuous-time Markov Chains (CTMCs)

A rate u_t is said to *generate* a **probability path** p_t if the time marginals of the associated CTMC are samples from p_t, *i.e.*, $X_t \sim p_t$.

Specifically, such a probability path satisfies the Kolmogorov Forward Equation (KFE):

$$\frac{\partial}{\partial t} p_t(x) = \sum_y u_t(x|y) p_t(y) = \sum_{y \neq x} u_t(x|y) p_t(y) - \sum_{y \neq x} u_t(y|x) p_t(x)$$



"The rate of change of a state's probability equals the net probability flux at that state."

Lipman *et al.*, Flow Matching Guide and Code, NeurIPS 2024 Tutorial

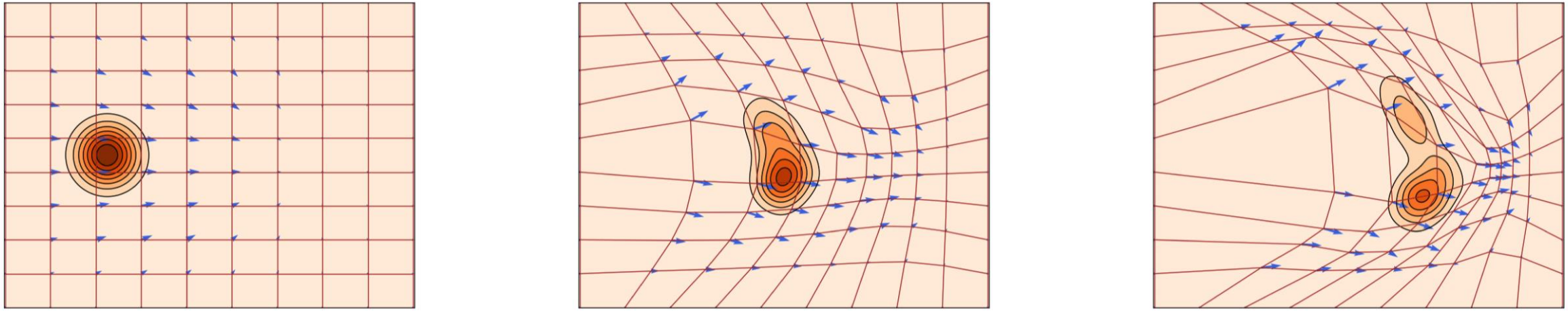
Discrete Flow Matching (DFM)

Consider a discrete space over sequences of fixed length N, $\mathcal{X} = \mathcal{T}^N$ where $\mathcal{T} = \{1, 2, \cdots, M\}$ is a vocabulary of size M.

The goal of DFM is to **learn the marginal rate** $u_t(x|x_t)$ and simulate transitions of a CTMC:

$$\mathbb{P}(X_{t+h} = x | X_t = x_t) = \delta_{x_t}(x) + h u_t(x|x_t) + o(h)$$

that generates a probability path p_t interpolating an easy-to-sample distribution and the data distribution.



Lipman *et al.*, Flow Matching Guide and Code, NeurIPS 2024 Tutorial

Discrete Flow Matching (DFM)

However, directly regressing the marginal rate is intractable, as it involves averaging over the data distribution. Instead, we can use the **conditional rate** as the regression target:

$$u_t(x|x_t, x_0, x_1)$$

which generates a **conditional probability path** $p_t(x|x_0, x_1)$ with $p_0(x|x_0, x_1) = \delta_{x_0}(x)$ and $p_1(x|x_0, x_1) = \delta_{x_1}(x)$.

Discrete Flow Matching (DFM)

However, directly regressing the marginal rate is intractable, as it involves averaging over the data distribution. Instead, we can use the **conditional rate** as the regression target:

$$u_t(x|x_t, x_0, x_1)$$

which generates a **conditional probability path** $p_t(x|x_0, x_1)$ with $p_0(x|x_0, x_1) = \delta_{x_0}(x)$ and $p_1(x|x_0, x_1) = \delta_{x_1}(x)$.

(x_0, x_1) is a sample from a **coupling distribution** $\pi(x_0, x_1)$, *i.e.*, $(x_0, x_1) \sim \pi(x_0, x_1)$, which, in its simplest form, is modeled as the product of two independent densities:

$$\pi(x_0, x_1) = p(x_0)q(x_1)$$

Discrete Flow Matching (DFM)

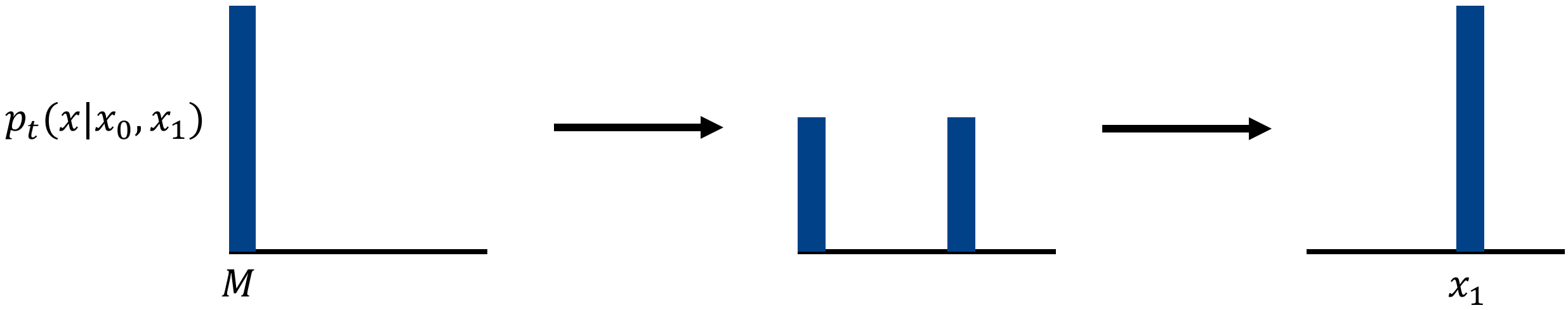
The marginal probability path and marginal rate can be obtained via simple averaging:

$$p_t(x) = \sum_{x_0, x_1} p_t(x|x_0, x_1)\pi(x_0, x_1)$$

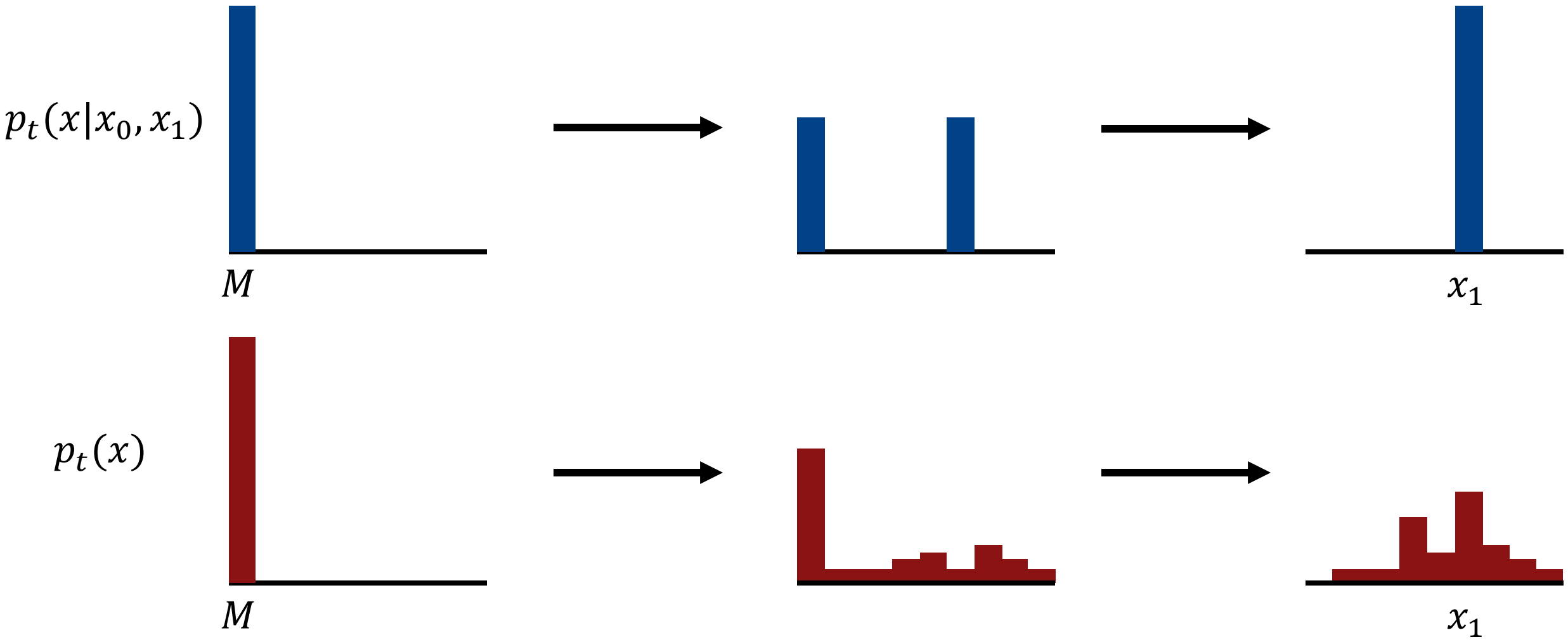
$$u_t(x|x_t) = \mathbb{E}_{p_t(x_0, x_1|x_t)} u_t(x|x_t, x_0, x_1)$$

where $p_0(x) = p(x)$ and $p_1(x) = q(x)$.

Discrete Flow Matching (DFM)



Discrete Flow Matching (DFM)



Discrete Flow Matching (DFM)

One popular target conditional probability path $p_t(x|x_0, x_1)$ is the **factorized token-wise probability path** [Gat *et al.*, NeurIPS 2024]:

$$p_t\left(x^i \middle| x_0^i, x_1^i\right) = (1 - \kappa_t)\delta_{x_0^i}\left(x^i\right) + \kappa_t \delta_{x_1^i}\left(x^i\right)$$

$$u_t\left(x^i \middle| x_t^i, x_0^i, x_1^i\right) = \frac{\dot{\kappa}_t}{1 - \kappa_t}\left(\delta_{x_1^i}\left(x^i\right) - \delta_{x_t^i}\left(x^i\right)\right)$$

where κ_t is a scheduler satisfying $\kappa_0 = 0$ and $\kappa_1 = 1$.

Discrete Flow Matching (DFM)

One popular target conditional probability path $p_t(x|x_0, x_1)$ is the **factorized token-wise probability path** [Gat *et al.*, NeurIPS 2024]:

$$p_t\left(x^i \middle| x_0^i, x_1^i\right) = (1 - \kappa_t)\delta_{x_0^i}\left(x^i\right) + \kappa_t \delta_{x_1^i}\left(x^i\right)$$

$$u_t\left(x^i \middle| x_t^i, x_0^i, x_1^i\right) = \frac{\dot{\kappa}_t}{1 - \kappa_t}\left(\delta_{x_1^i}\left(x^i\right) - \delta_{x_t^i}\left(x^i\right)\right)$$

where κ_t is a scheduler satisfying $\kappa_0 = 0$ and $\kappa_1 = 1$.

The marginal distributions of (corrupted) sequences are modeled as:

$$p_t(x|x_0, x_1) = \prod_{i=1}^{N} p_t(x^i|x_0^i, x_1^i)$$

that are characterized by the conditional rate:

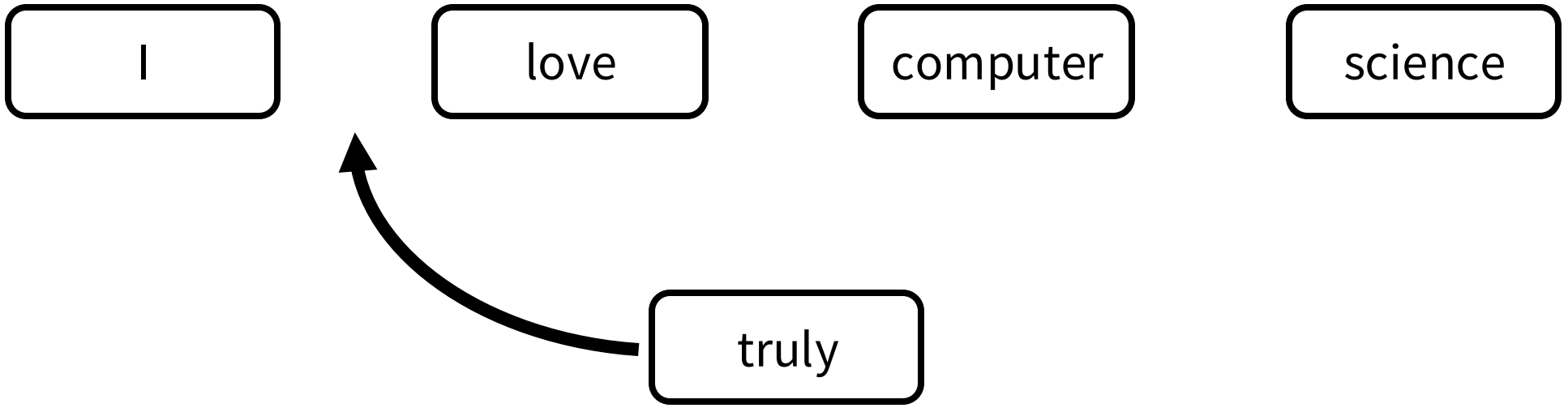
$$u_t(x|x_t, x_0, x_1) = \sum_i \delta_{x_t}(x^{\neg i}) u_t(x^i|x_t^i, x_0^i, x_1^i) \quad \text{with } \delta_{x_t}\left(x^{\neg i}\right) = \prod_{j \neq i} \delta_{x_t^j}\left(x^j\right)$$

Discrete Flow Matching (DFM)

However, DFM has difficulty generating sequences of varying lengths, as illustrated in the example below.

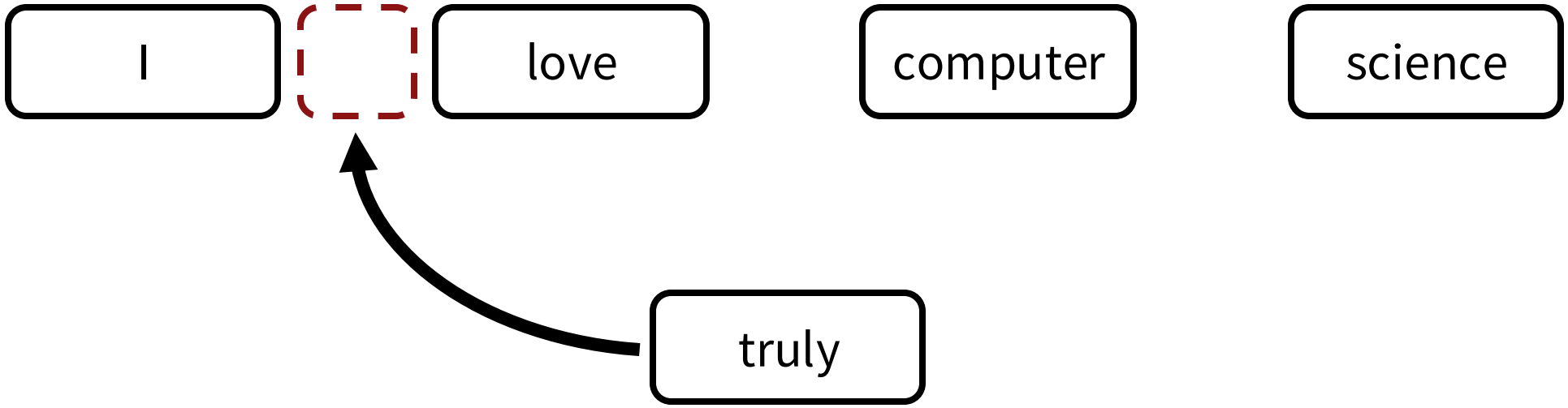
Discrete Flow Matching (DFM)

However, DFM has difficulty generating sequences of varying lengths, as illustrated in the example below.



Discrete Flow Matching (DFM)

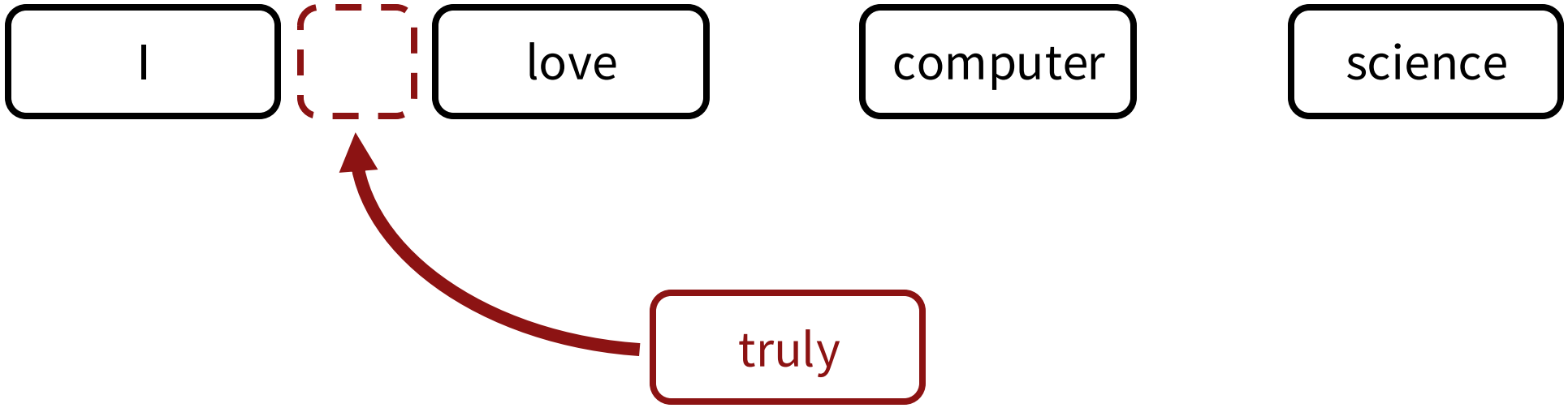
However, DFM has difficulty generating sequences of varying lengths, as illustrated in the example below.



Where to put a new token?

Discrete Flow Matching (DFM)

However, DFM has difficulty generating sequences of varying lengths, as illustrated in the example below.

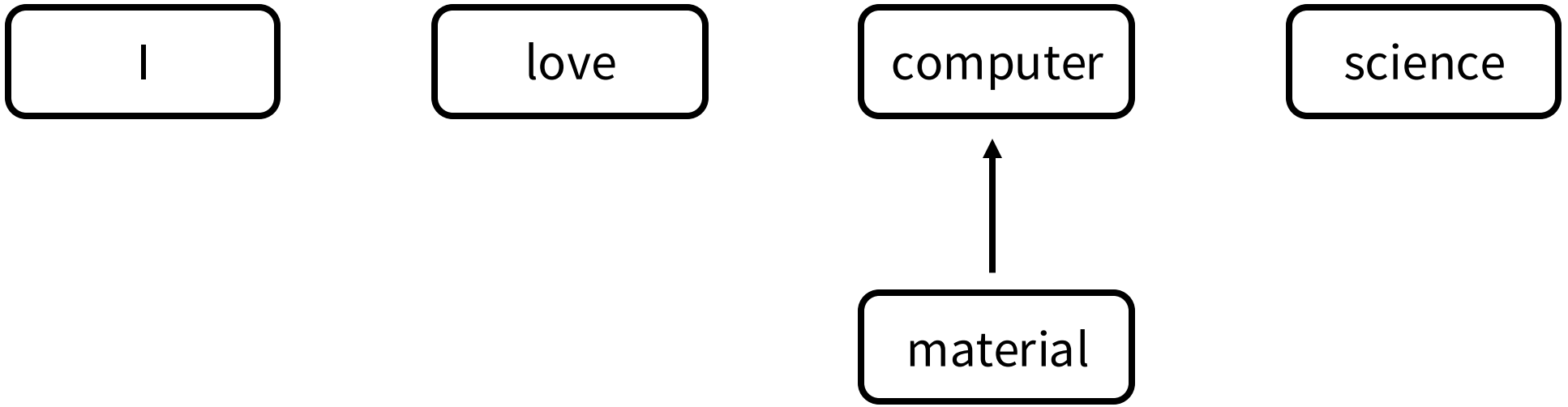


Where to put a new token?

Which token to put?

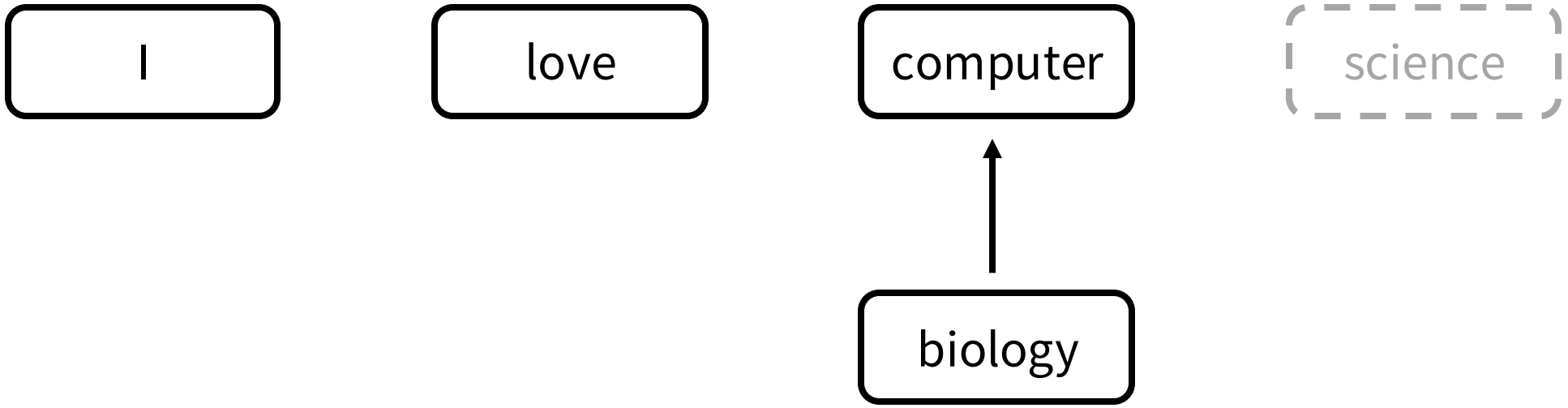
Discrete Flow Matching (DFM)

However, DFM has difficulty generating sequences of varying lengths, as illustrated in the example below.



Discrete Flow Matching (DFM)

However, DFM has difficulty generating sequences of varying lengths, as illustrated in the example below.

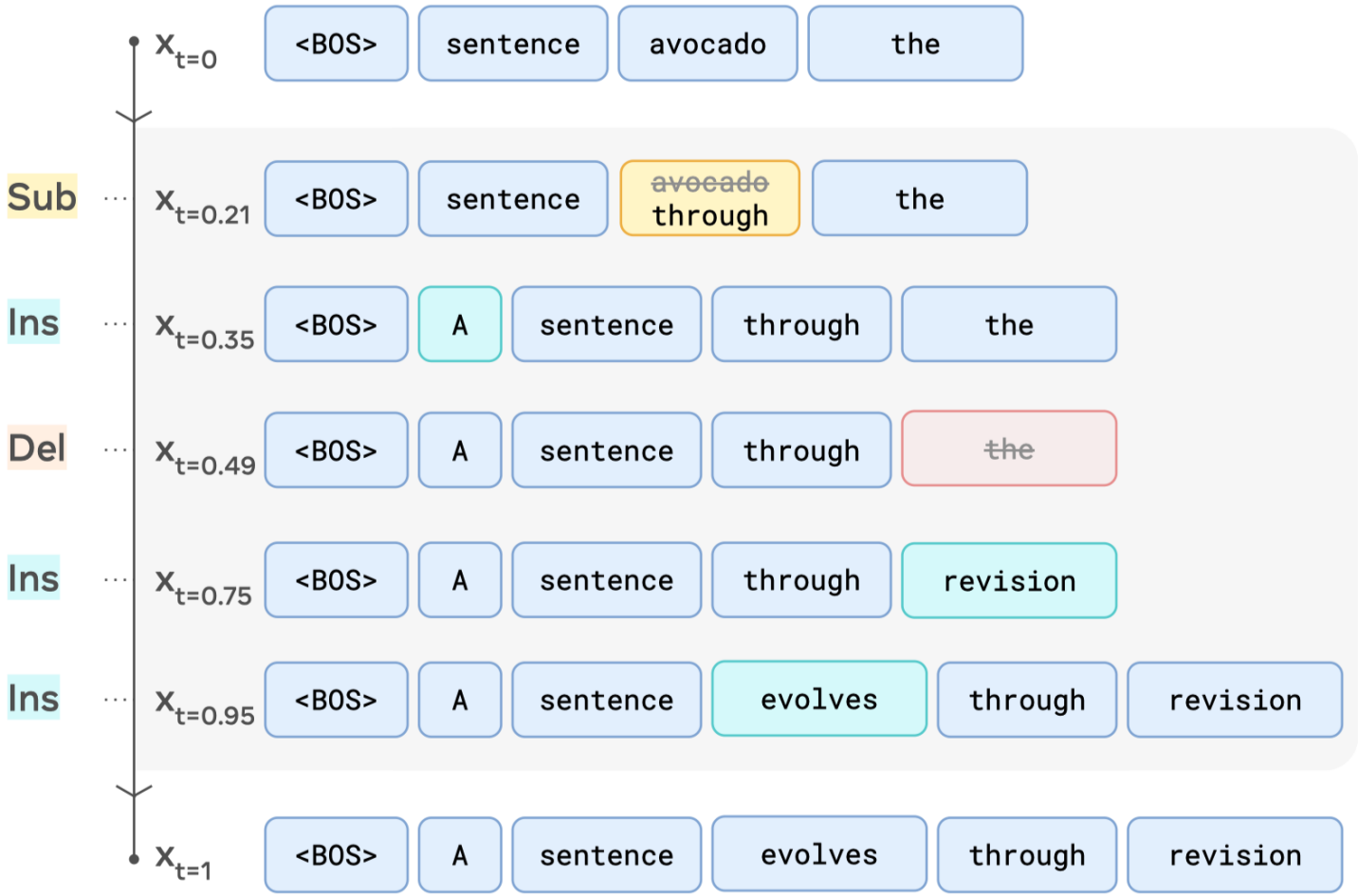


Edit Flows

In this work, we extend the DFM framework by allowing two sequences $x, x_t \in \bigcup_{n=0}^{N} \mathcal{T}^n$ differ by one *edit operation*, which is one of *insertion*, *deletion*, and *substitution*:

- $\text{ins}(x, i, a) = (x^1, \ldots, x^i, a, x^{i+1}, \ldots, x^{n(x)})$
- $\text{del}(x, i) = \left(x^1, \ldots, x^{i-1}, x^{i+1}, \ldots x^{n(x)}\right)$
- $\text{sub}(x, i, a) = (x^1, \ldots, x^{i-1}, a, x^{i+1}, \ldots, x^{n(x)})$

Edit Flows



Edit Flow sampling process.

Edit Flows

Since insertions, deletions, and substitutions produce mutually exclusive outcomes, we can define separate rates for each operation:

- $u_t^\theta(\text{ins}(x, i, a)|x) = \lambda_{t,i}^{\text{ins}}(x) Q_{t,i}^{\text{ins}}(a|x)$ for $i \in \{1, \dots, n(x)\}$
- $u_t^\theta(\text{del}(x, i)|x) = \lambda_{t,i}^{\text{del}}(x)$ for $i \in \{1, \dots, n(x)\}$
- $u_t^\theta(\text{sub}(x, i, a)|x) = \lambda_{t,i}^{\text{sub}}(x) Q_{t,i}^{\text{sub}}(a|x)$ for $i \in \{1, \dots, n(x)\}$

Edit Flows

Since insertions, deletions, and substitutions produce mutually exclusive outcomes, we can define separate rates for each operation:

- $u_t^\theta(\text{ins}(x, i, a)|x) = \lambda_{t,i}^{\text{ins}}(x) Q_{t,i}^{\text{ins}}(a|x)$ for $i \in \{1, \ldots, n(x)\}$
- $u_t^\theta(\text{del}(x, i)|x) = \lambda_{t,i}^{\text{del}}(x)$ for $i \in \{1, \ldots, n(x)\}$
- $u_t^\theta(\text{sub}(x, i, a)|x) = \lambda_{t,i}^{\text{sub}}(x) Q_{t,i}^{\text{sub}}(a|x)$ for $i \in \{1, \ldots, n(x)\}$

The total rates of editing operations

"Which operation should be applied to which token in x?"

Edit Flows

Since insertions, deletions, and substitutions produce mutually exclusive outcomes, we can define separate rates for each operation:

- $u_t^\theta(\text{ins}(x, i, a)|x) = \lambda_{t,i}^{\text{ins}}(x) Q_{t,i}^{\text{ins}}(a|x)$ for $i \in \{1, \dots, n(x)\}$
- $u_t^\theta(\text{del}(x, i)|x) = \lambda_{t,i}^{\text{del}}(x)$ for $i \in \{1, \dots, n(x)\}$
- $u_t^\theta(\text{sub}(x, i, a)|x) = \lambda_{t,i}^{\text{sub}}(x) Q_{t,i}^{\text{sub}}(a|x)$ for $i \in \{1, \dots, n(x)\}$

Normalized distributions over token values $a \in \mathcal{T}$

"Which value should be added or used to replace the selected token?"

Edit Flows

Indeed, Edit Flow is a generalization of existing techniques, each of which can be obtained by restricting the rates.

- **DFM**: A special case that permits only substitutions, while disallowing token insertions and deletions (*i.e.*, $\lambda_{t,i}^{\text{ins}} = 0$ and $\lambda_{t,i}^{\text{del}} = 0$) → **Lacks support for varying lengths**;
- **AR**: A special case that permits insertions to occur at the rightmost location (*i.e.*, $\lambda_{t,n(x)}^{\text{ins}} \neq 0$ → **Incapable of making corrections**.

Edit Flows are powerful.
But how can we train them?

Learning Edit Flows via Augmented CTMCs

The main challenge lies in defining the conditional probability path $p_t(x|x_0, x_1)$ and its rate $u_t(x|x_0, x_1)$ that our model can regress due to **the enormous number of possible transitions** that take one sequence to another:

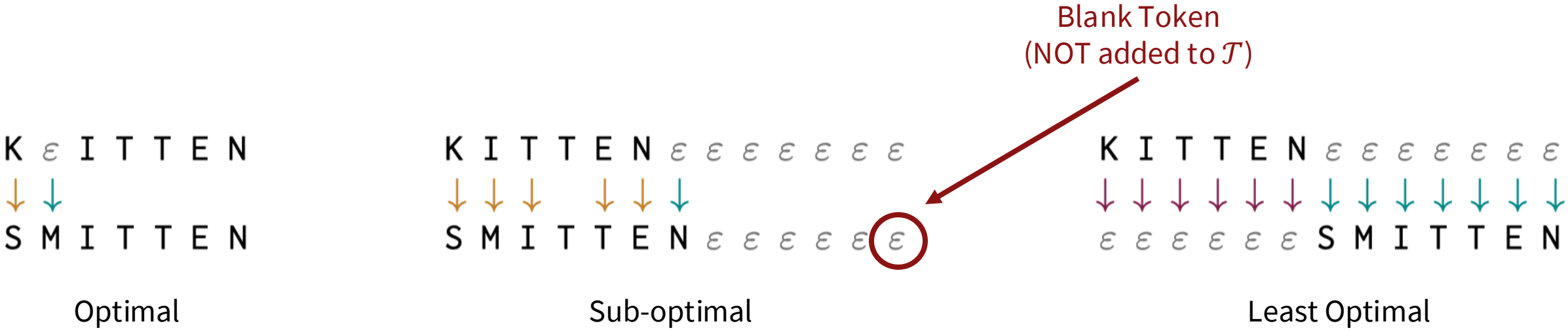
Optimal

Sub-optimal

Least Optimal

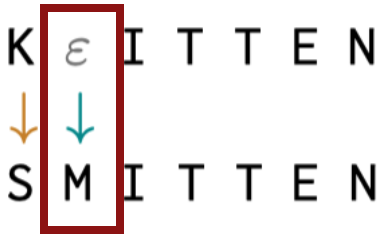
Learning Edit Flows via Augmented CTMCs

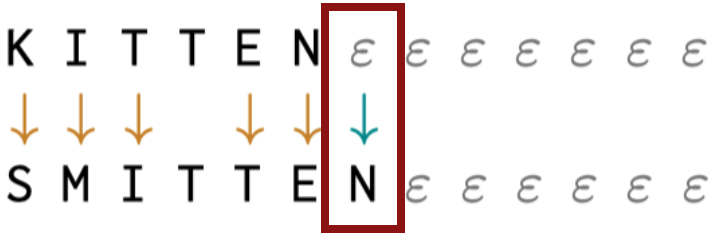
The key idea for simplifying the problem is to **augment the space** $\mathcal{X} = \bigcup_{n=0}^{N} \mathcal{T}^n$ by introducing an auxiliary token ε and define an auxiliary Markov process via **alignment**.

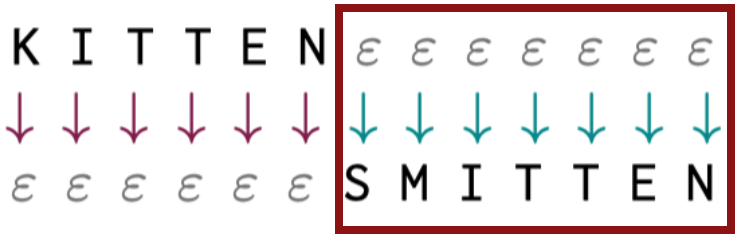


Learning Edit Flows via Augmented CTMCs

Edit operations can be recovered as tuples $(a \to b)$ with $a, b \in \mathcal{T} \cup \{\varepsilon\}$.



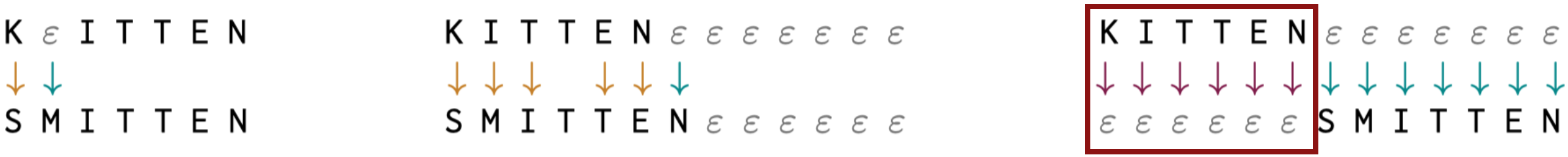




Insertion: $a = \varepsilon$

Learning Edit Flows via Augmented CTMCs

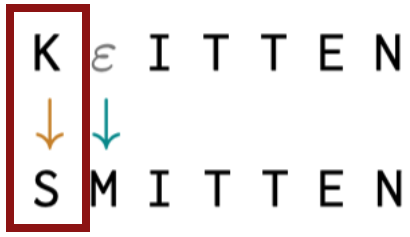
Edit operations can be recovered as tuples $(a \to b)$ with $a, b \in \mathcal{T} \cup \{\varepsilon\}$.

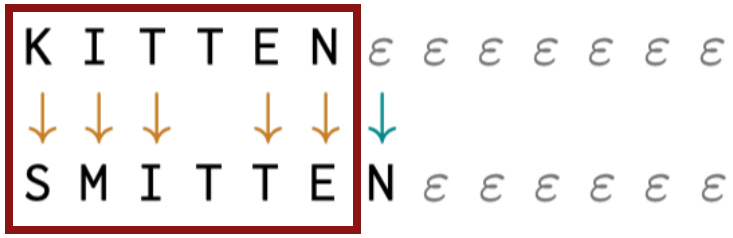


Deletion: $\mathrm{b} = \varepsilon$

Learning Edit Flows via Augmented CTMCs

Edit operations can be recovered as tuples $(a \to b)$ with $a, b \in \mathcal{T} \cup \{\varepsilon\}$.





Substitution: a $\neq \varepsilon$ and b $\neq \varepsilon$

Learning Edit Flows via Augmented CTMCs

Formally, let

- $\mathcal{X} = \bigcup_{n=0}^{N} \mathcal{T}^n$: The space of normal (raw) sequences of different lengths;
- $\mathcal{Z} = (\mathcal{T} \cup \{\varepsilon\})^N$: The space of aligned sequences (*i.e.*, padded with ε's);
- $f_{\text{rm}-\text{blanks}}\ (\mathcal{Z} \to \mathcal{X})$: An operation that strips away all ε tokens;
  - K$\underline{\varepsilon}$ITTEN → KITTEN
  - KITTEN$\underline{\varepsilon\varepsilon\varepsilon}$ → KITTEN
  - K$\underline{\varepsilon}$I$\underline{\varepsilon}$T$\underline{\varepsilon}$T$\underline{\varepsilon}$E$\underline{\varepsilon}$N → KITTEN

Learning Edit Flows via Augmented CTMCs

Given samples from the source $x_0 \sim p(x)$ and target $x_1 \sim q(x)$ in $\mathcal{X}$, one can construct aligned sequences z_0 and z_1 by:

- randomly padding the sequences;
- solving for the optimal alignment that corresponds to the minimal edit distance.

This yields a coupling $\pi(z_0, z_1)$ over the auxiliary variables satisfying

$$p(x) = \sum_{z_0} \sum_{z_1} \pi(z_0, z_1) \delta_{f_{\text{rm-blanks}(z_0)}}(x)$$

$$q(x) = \sum_{z_0} \sum_{z_1} \pi(z_0, z_1) \delta_{f_{\text{rm-blanks}(z_1)}}(x)$$

Learning Edit Flows via Augmented CTMCs

Furthermore, for $z_0, z_1 \sim \pi$, we define a conditional probability path over the augmented space $\mathcal{X} \times \mathcal{Z}$:

$$p_t(x, z | x_0, z_0, x_1, z_1) = p_t(x, z | z_0, z_1) = p_t(z | z_0, z_1) \delta_{f_{\text{rm-blanks}(z)}}(x)$$

where $p_t(z | z_0, z_1)$ is a token-wise mixture probability path:

$$p_t(z | z_0, z_1) = \prod_{i=1}^{N} p_t(z^i | z_0^i, z_1^i)$$

Learning Edit Flows via Augmented CTMCs

The conditional rate $u_t(x, z|x_t, z_t, z_0, z_1)$ corresponding to the probability path is:

$$u_t(x, z|x_t, z_t, z_0, z_1) = \delta_{f_{\text{rm-blanks}(z)}}(x) \sum_{i=1}^{N} \frac{\dot{\kappa}_t}{1-\kappa_t} \left(\delta_{z_1^i}(z^i) - \delta_{z_t^i}(z^i) \right) \delta_{z_t}(z^{\neg i})$$

which is analogous to the conditional rate discussed in DFM:

$$u_t(x|x_t, x_0, x_1) = \sum_i \frac{\dot{\kappa}_t}{1-\kappa_t} \left(\delta_{x_1^i}(x^i) - \delta_{x_t^i}(x^i) \right) \delta_{x_t}(x^{\neg i})$$

Learning Edit Flows via Augmented CTMCs

By marginalizing the conditional rate $u_t(x, z|x_t, z_t, z_0, z_1)$, we obtain the unconditional rate of a CTMC over the original state space $\mathcal{X}$:

$$u_t(x|x_t) = \sum_z \mathbb{E}_{p_t(z_0, z_1, z_t|x_t)} u_t(x, z|x_t, z_t, z_0, z_1)$$

Learning Edit Flows via Augmented CTMCs

It is proved that this unconditional rate can be learned by minimizing the training loss of form:

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{\pi(z_0,z_1) \\ t,p_t(x_t,z_t|z_0,z_1)}} \left[\sum_{x \neq x_t} u_t^\theta(x|x_t) - \sum_{i=1}^{N} \mathbf{1}_{[z_1^i \neq z_t^i]} \frac{\dot{\kappa}_t}{1-\kappa_t} \log u_t^\theta(x(z_t, i, z_1^i)|x_t) \right]$$

where $x(z_t, i, z_1^i) = f_{\text{rm}-\text{blanks}}(z_t^1, \ldots, z_t^{i-1}, z_1^i, z_t^{i+1}, \ldots, z_t^N)$ is an output of one of the possible edit operations (insertion, deletion, and substitution).

Learning Edit Flows via Augmented CTMCs

It is proved that this unconditional rate can be learned by minimizing the training loss of form:

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{\pi(z_0,z_1) \\ t,p_t(x_t,z_t|z_0,z_1)}} \left[\sum_{x \neq x_t} u_t^\theta(x|x_t) - \sum_{i=1}^{N} \mathbf{1}_{[z_1^i \neq z_t^i]} \frac{\dot{\kappa}_t}{1-\kappa_t} \log u_t^\theta(x(z_t, i, z_1^i)|x_t) \right]$$

Regularize spurious editing operations

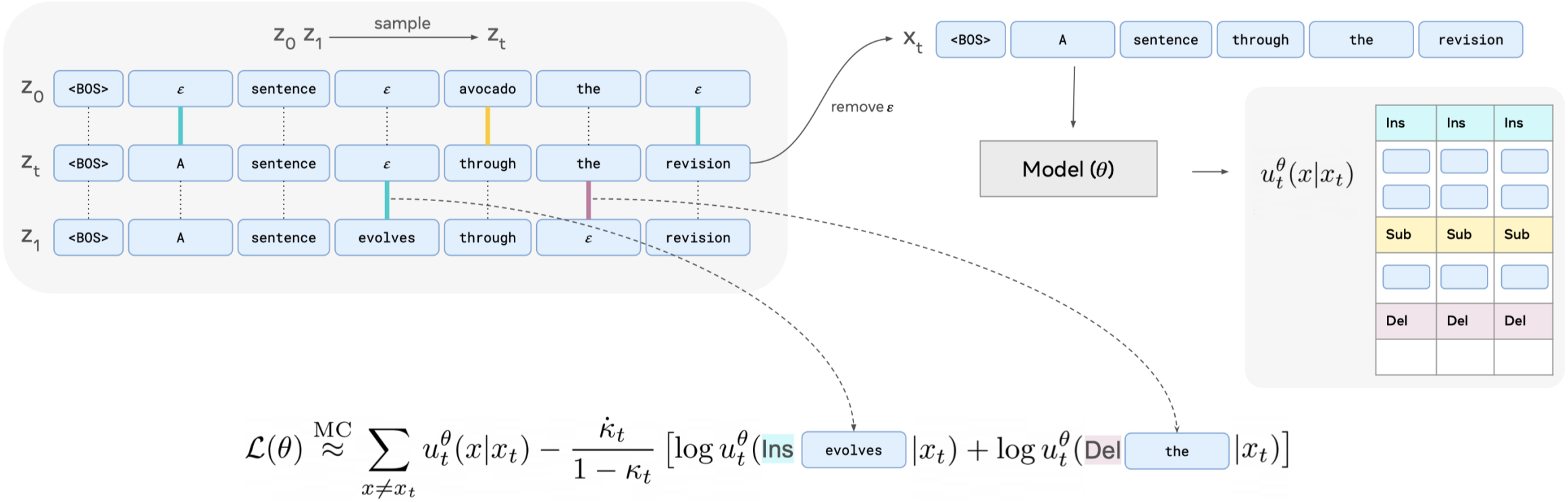
Learning Edit Flows via Augmented CTMCs

It is proved that this unconditional rate can be learned by minimizing the training loss of form:

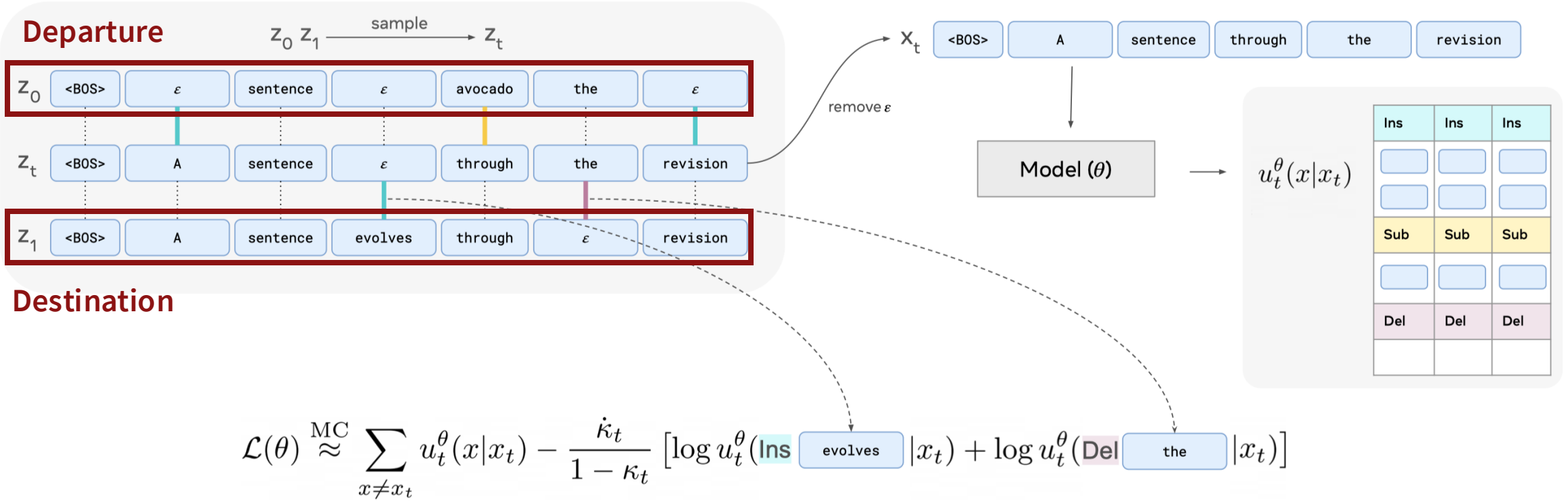
$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{\pi(z_0,z_1) \\ t,p_t(x_t,z_t|z_0,z_1)}} \left[\sum_{x \neq x_t} u_t^\theta(x|x_t) - \sum_{i=1}^{N} \mathbf{1}_{[z_1^i \neq z_t^i]} \frac{\dot{\kappa}_t}{1-\kappa_t} \log u_t^\theta(x(z_t, i, z_1^i)|x_t) \right]$$

Weighted cross-entropy over edit operations for $x_t \to x_1$

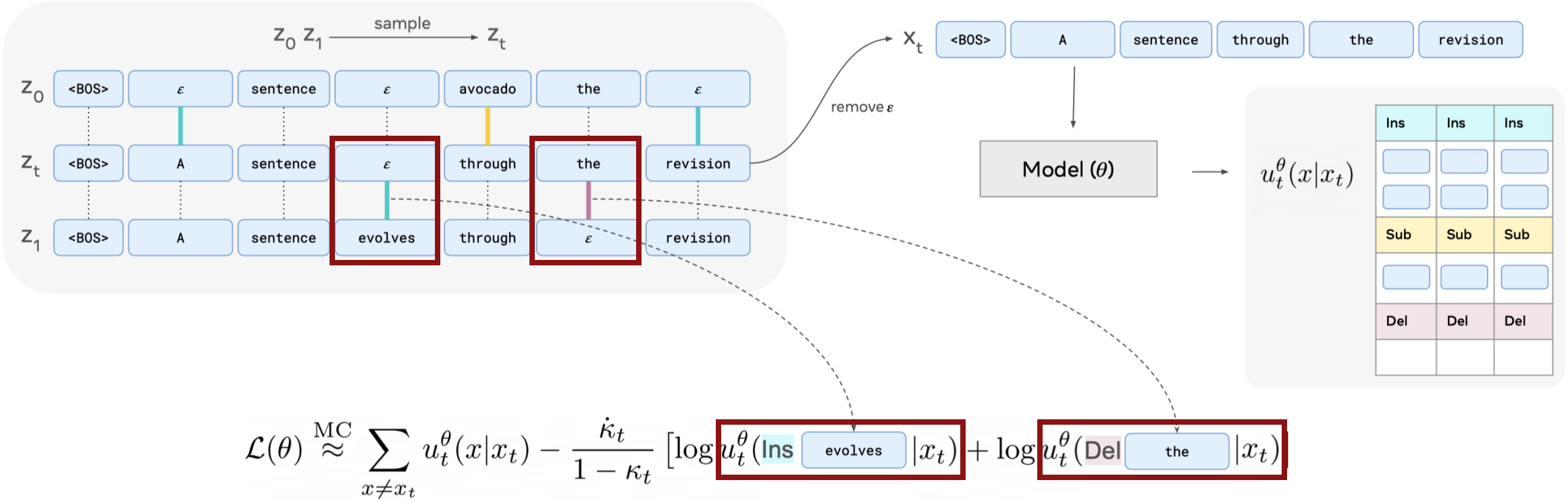
Learning Edit Flows via Augmented CTMCs



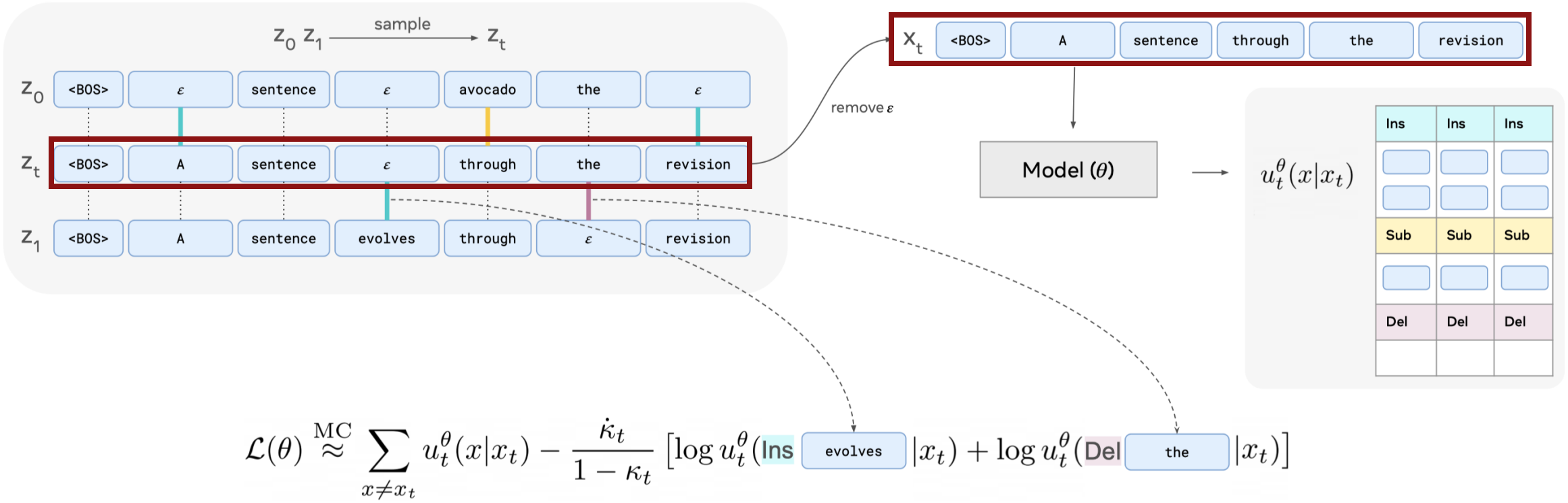
Learning Edit Flows via Augmented CTMCs



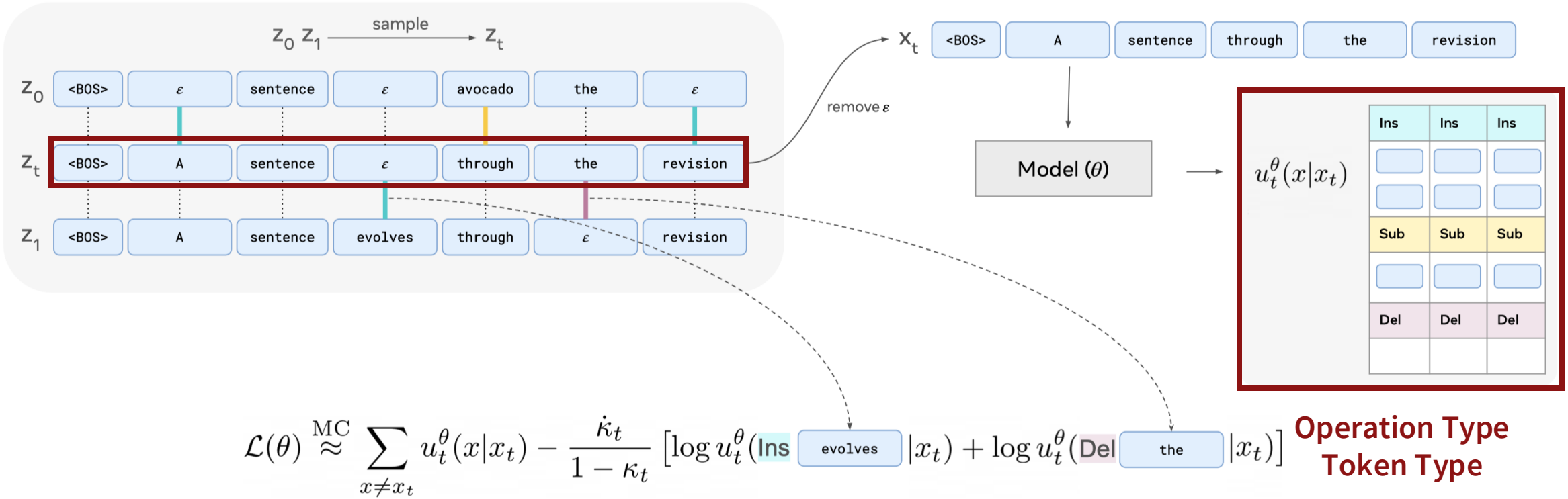
Learning Edit Flows via Augmented CTMCs



Learning Edit Flows via Augmented CTMCs



Learning Edit Flows via Augmented CTMCs



Learning Edit Flows via Augmented CTMCs

It is proved that this unconditional rate can be learned by minimizing the training loss of form:

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{\pi(z_0,z_1) \\ t,p_t(x_t,z_t|z_0,z_1)}} \left[\sum_{x \neq x_t} u_t^\theta(x|x_t) - \sum_{i=1}^{N} \mathbf{1}_{[z_1^i \neq z_t^i]} \frac{\dot{\kappa}_t}{1-\kappa_t} \log u_t^\theta(x(z_t, i, z_1^i)|x_t) \right]$$

where $x(z_t, i, z_1^i) = f_{\text{rm}-\text{blanks}}(z_t^1, \ldots, z_t^{i-1}, z_1^i, z_t^{i+1}, \ldots, z_t^N)$ is an output of one of the possible edit operations (insertion, deletion, and substitution).

Theorem 3.1 (Flow Matching with Auxiliary Processes). *Let $u_t(x, z|x_t, z_t)$ be a rate over the augmented space of $\mathcal{X} \times \mathcal{Z}$ that generates $p_t(x, z)$, then*

$$u_t(x|x_t) \triangleq \sum_z \mathbb{E}_{p_t(z_t|x_t)} u_t(x, z|x_t, z_t) \qquad \textit{generates} \qquad p_t(x) \triangleq \sum_z p_t(x, z), \tag{17}$$

and furthermore, for any Bregman divergence $D_\phi(a, b) = \phi(a) - \phi(b) - \langle a - b, \frac{\mathrm{d}}{\mathrm{d}b}\phi(b) \rangle$ defined by a convex function ϕ, we have that

$$\frac{\mathrm{d}}{\mathrm{d}\theta} \mathbb{E}_{x_t, z_t \sim p_t(x,z)} D_\phi \left(\sum_z u_t(\cdot, z|x_t, z_t), u_t^\theta(\cdot|x_t) \right) = \frac{\mathrm{d}}{\mathrm{d}\theta} \mathbb{E}_{x_t \sim p_t(x)} D_\phi \left(u_t(\cdot|x_t), u_t^\theta(\cdot|x_t) \right). \tag{18}$$

Sampling Edit Flows

To sample from the model, we transport a source sample $x_0 \sim p$ to time $t = 1$, by simulating the CTMC defined with the learned rate $u_t^\theta(x|x_t)$. The exact probability of an edit operation characterized by the rate $\lambda_{t,i}$ occurring within $(t, t+h)$ is:

$$1 - e^{-\int_t^{t+h} \lambda_{t,i}(X_t)dt} \approx h\lambda_{t,i}(X_t) \qquad \text{(Taylor Expansion)}$$

Sampling Edit Flows

As with previous discrete diffusion/flow models, each next states are sampled **independently**. The sampling proceeds as follows:

1. For each position i, sample whether to insert with probability $h\lambda_{t,i}^{\text{ins}}(X_t)$ and whether to delete or substitute with probability $h\left(\lambda_{t,i}^{\text{del}}(X_t) + \lambda_{t,i}^{\text{sub}}(X_t)\right)$;
2. If insertion or substitution occurs at token i, sample the new token value from $Q_{t,i}^{\text{ins}}(a|X_t)$ or $Q_{t,i}^{\text{sub}}(a|X_t)$;
3. Move to the next timestep: $t \leftarrow t + h$.

Sampling Edit Flows

Following [Nisonoff *et al*., 2024], the authors apply classifier-free guidance (CFG) [Ho and Salimans, 2021] during sampling, by adjusting the predicted rates:

$$\tilde{u}_t(x|x_t, c) = \tilde{\lambda}_{t,i}(x_t, c)\tilde{Q}_t(a|x_t, c)$$

where $\tilde{\lambda}_t(x_t, c) = \lambda_{t,i}(x_t|c)^{1+w}\lambda_{t,i}(x_t)^{-w}$ and w is guidance strength.

Experiments

Edit Flow is mainly tested on text generation tasks, including:

1. Image-to-Text Generation (280M models);
2. Text Generation (1.3B models);
3. Code Generation (1.3B models).

Experiments

As baselines, the authors consider SotA AR/non-AR models:

1. Llama 2 [Touvron *et al.*, 2023]
2. Masked DFM [Gat *et al.*, 2024]

These baselines are compared against three variants of Edit Flow:

1. **Default Model**: Starts from $p = \delta_{\emptyset}$ (an empty string) and inserts/delete tokens;
2. **Uniform $\mathbf{X_0}$ + Edit Flow**: $X_0 = (X^1, \ldots, X^{100})$ where $X^i \sim p_{\text{emp}}$ using empirical training token distribution;
3. **Localized Edit Flow**: A variant of edit flow using a localized propagation process.

B.1 Localized propagation paths

Edit Flows leverage an underlying conditional probability path $p_t(z|z_0, z_1)$ and associated rates $u_t(z|z_t)$, so far given by the factorized token-wise mixture. Let us further generalize this probability path and associated rate to be non-factorized, applying auxiliary variables again. We first re-express this probability path through an auxiliary boolean variable $\mathsf{m} \in \{\texttt{false}, \texttt{true}\}^N$:

$$p_t(z|z_0, z_1) = \sum_{\mathsf{m}} p_t(\mathsf{m}|z_0, z_1) p_t(z|\mathsf{m}, z_0, z_1), \tag{37}$$

$$\text{where } p_t(z|\mathsf{m}, z_0, z_1) = \prod_{i=1}^{N} \mathbb{1}_{[\neg \mathsf{m}^i]} \delta_{z_0^i}(z^i) + \mathbb{1}_{[\mathsf{m}^i]} \delta_{z_1^i}(z^i), \tag{38}$$

Experiments

All baselines and variants of Edit Flow share the architecture from Llama [Grattafiori *et al.*, 2024; Touvron *et al.*, 2023], with 280M and 1.3B parameters.

- All models are trained using the same compute budget;
- The **maximum sequence length** during training is set to **1024** tokens;
- AR baselines use causal attention, while non-AR methods full self-attention;
- For Edit Flow, FlexAttention [Dong *et al.*, 2024] is used to handle varying lengths.

Experiments

For image captioning tasks, all methods are trained on the MS COCO dataset [Lin *et al.*, 2014] and Image Captioning 3M dataset.

Method	MS COCO			Image Captioning 3M		
	METEOR	CIDEr	SPICE	ROUGE-L	CIDEr	SPICE
VLP† (Zhou et al., 2020)	28.4	117.7	21.3	24.3	77.5	16.5
ClipCap† (Mokady et al., 2021)	27.1	108.3	20.1	26.7	87.2	18.5
Autoregressive	25.7	95.5	19.6	25.2	85.8	17.8
Mask DFM	25.3	95.6	19.2	27.4	96.2	20.3
Edit Flow (**Ours**)	27.4	108.1	21.1	29.0	101.9	21.7
Localized Edit Flow (**Ours**)	27.4	105.1	22.1	28.3	99.7	20.8

Experiments

For text benchmarks, all methods are trained on the DCLM baseline 1.0 [Li *et al.*, 2024] dataset. Edit Flow bridges the gap between non-AR- and AR-based methods.

Method	HellaSwag	ARC-E	ARC-C	PIQA	OBQA	WinoGrande
Autoregressive	49.5	71.0	36.3	76.0	30.4	62.1
Mask DFM	38.3	55.4	27.8	65.3	22.6	52.3
Edit Flow (CFG applied to u_t) (**Ours**)	49.0	63.1	33.0	68.8	28.6	53.6
Edit Flow (CFG applied to $\mathcal{L}$) (**Ours**)	54.5	61.0	34.0	65.0	37.2	54.3

Experiments

For code generation, all methods are trained on the CodeLlama datamix [Roziere *et al.*, 2023]. *Oracle Length* denotes the case where GT code length is known during generation.

	Method	HumanEval		HumanEval+		MBPP	
		Pass@1	Pass@10	Pass@1	Pass@10	Pass@1	Pass@10
	Autoregressive (Gat et al., 2024)	14.3	21.3			17.0	34.3
	Autoregressive†	17.0	34.7	14.0	28.6	25.6	45.4
Non-AR	Mask DFM (Gat et al., 2024)	6.7	13.4			6.7	20.6
Non-AR	Mask DFM (Oracle Length) (Gat et al., 2024)	11.6	18.3			13.1	28.4
Non-AR	Mask DFM†	9.1	17.6	7.9	13.4	6.2	25.0
Non-AR	Uniform X_0 + Edit Flow (**Ours**)	9.7	24.3	9.7	19.5	9.4	33.4
Non-AR	Edit Flow (**Ours**)	12.8	24.3	10.4	20.7	10.0	36.4
Non-AR	Localized Edit Flow (**Ours**)	14.0	22.6	10.4	18.9	14.8	34.0

Conclusion

To summarize, this paper presents:

- Edit Flow, a novel extension of Discrete Flow Matching [Gat *et al.*, 2024], which enables variable-length generation via edit operations-insertion, deletion, and substitution;
- A novel CTMC with conditional rates defined over an augmented state space, enabling a tractable training objective for Edit Flow;
- Extensive evaluations in image captioning and text/code generation demonstrate that non-autoregressive models can serve as potential replacements for current AR models.

Edit Flows: Flow Matching with Edit Operations

Insert-only

```
def is_prime(n: int) -> bool:
```

Insert + Delete + Substitute

```
def is_prime(n: int) -> bool:
    if n <= 1:
        return True
    for i in range(2, n):
        if i % n == 0:
            return True
    return False
```

Seungwoo Yoo
KAIST
https://dvelopery0115.github.io