A Continuous-Time Markov Chain Framework for Insertion Language Models

Dhruvesh Patel[*] **Benjamin Rozonoyer**[*] **Soumitra Das**[*]

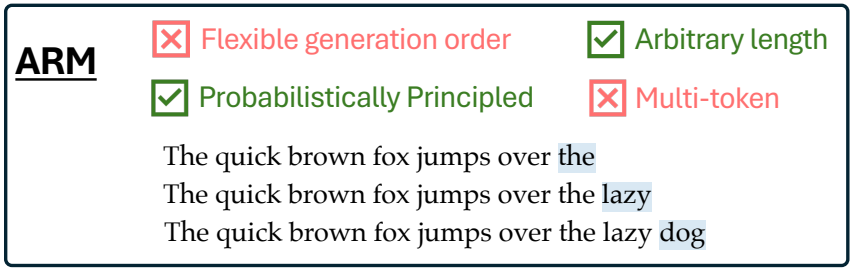
Tahira Naseem[†] **Tim G. J. Rudner**[‡§] **Andrew McCallum**[*]

AISTATS 2026

UMass Amherst[*] IBM Research[†] University of Toronto[‡] Vijil[§]

Why insertions?

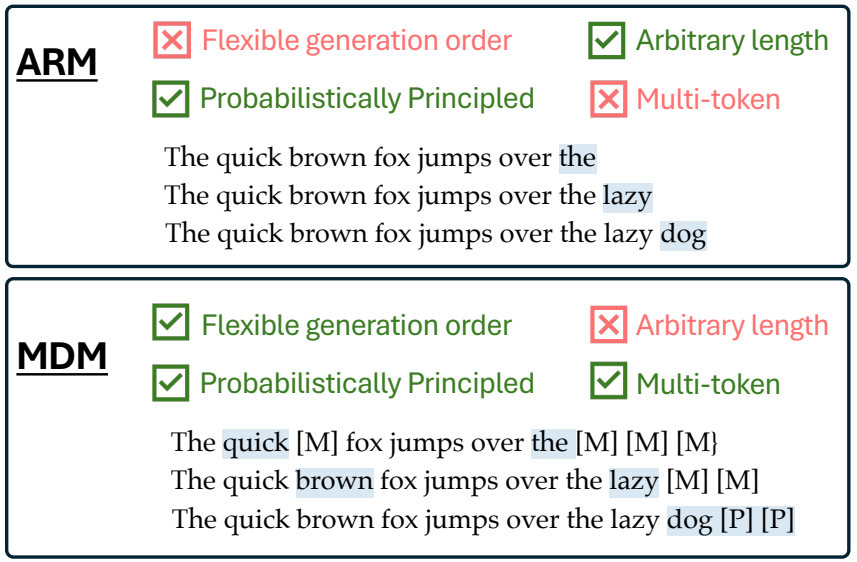
ARMs hard-code a left-to-right ordering assumption. Not suitable for some tasks.



Why insertions?

ARMs hard-code a left-to-right ordering assumption. Not suitable for some tasks.

MDMs relax generation order. Variable-length generation is non-trivial.

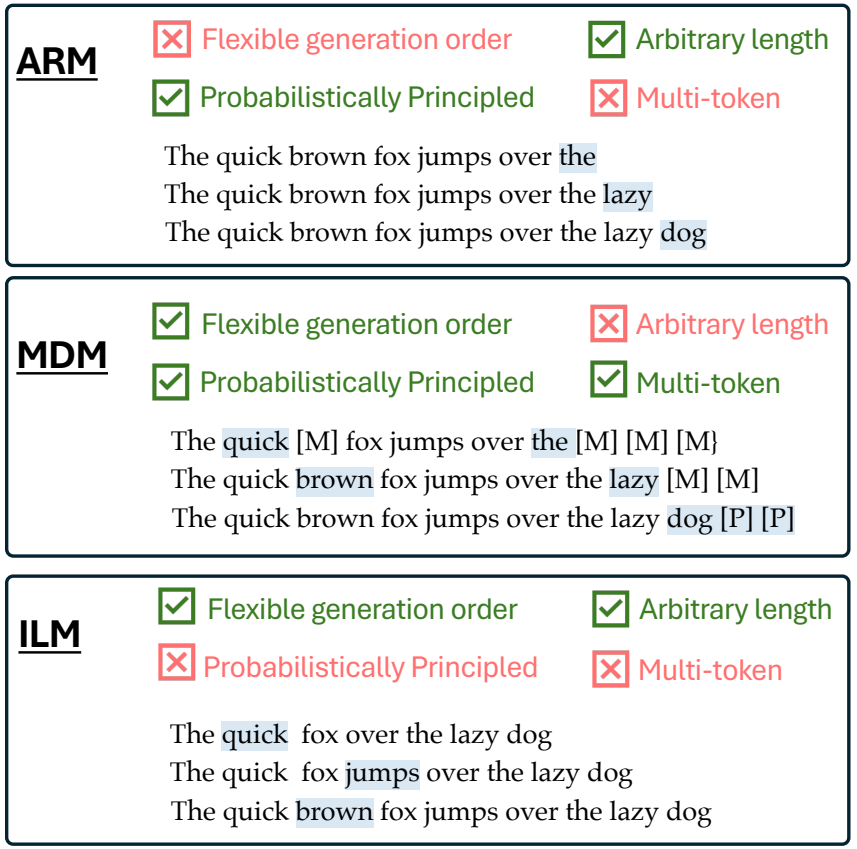


Why insertions?

ARMs hard-code a left-to-right ordering assumption. Not suitable for some tasks.

MDMs relax generation order. Variable-length generation is non-trivial.

ILMs have the right flexibility, but existing objectives and sampling rules are ad hoc.



Principled Insertion-based Generation

ARMs hard-code a left-to-right ordering assumption. Not suitable for some tasks.

MDMs relax generation order. Variable-length generation is non-trivial.

ILMs have the right flexibility, but existing objectives and sampling rules are ad hoc.

DILMs have diffusion-style denoising objective and sampling to combine all the benefits.

ARM

- ☒ Flexible generation order
- ☑ Arbitrary length
- ☑ Probabilistically Principled
- ☒ Multi-token

The quick brown fox jumps over the
The quick brown fox jumps over the lazy
The quick brown fox jumps over the lazy dog

MDM

- ☑ Flexible generation order
- ☒ Arbitrary length
- ☑ Probabilistically Principled
- ☑ Multi-token

The quick [M] fox jumps over the [M] [M] [M}
The quick brown fox jumps over the lazy [M] [M]
The quick brown fox jumps over the lazy dog [P] [P]

ILM

- ☑ Flexible generation order
- ☑ Arbitrary length
- ☒ Probabilistically Principled
- ☒ Multi-token

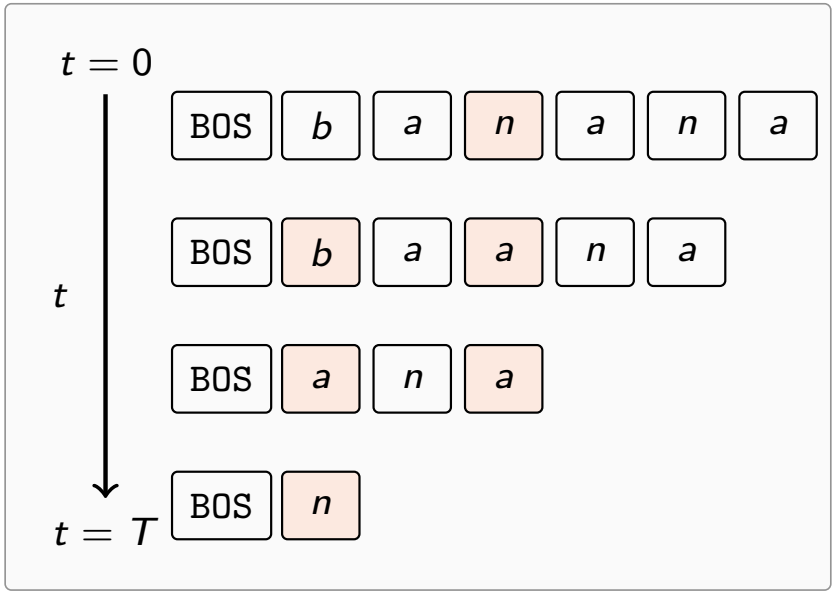
The quick fox over the lazy dog
The quick fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog

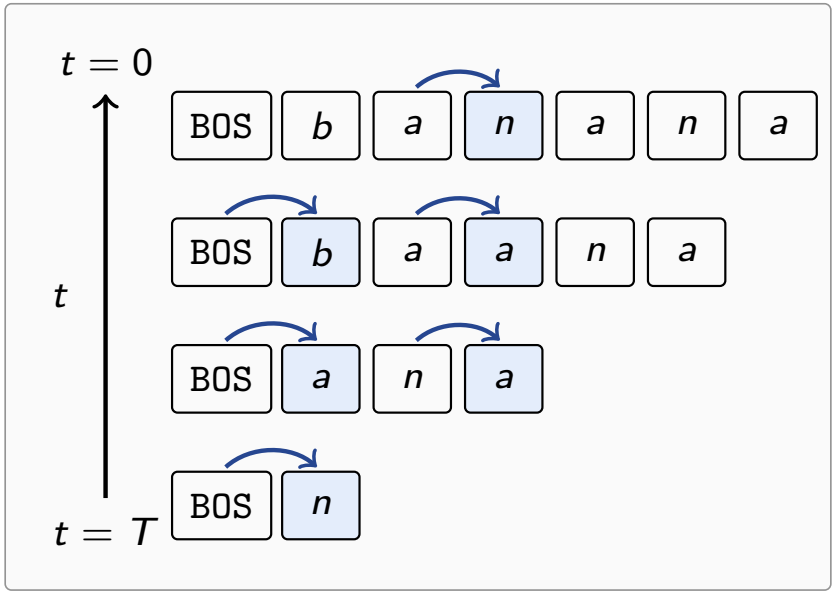
DILM

- ☑ Flexible generation order
- ☑ Arbitrary length
- ☑ Probabilistically Principled
- ☑ Multi-token

The quick fox over the dog
The quick fox jumps over the dog
The quick brown fox jumps over the lazy dog

A CTMC framework for Insertion Language Models

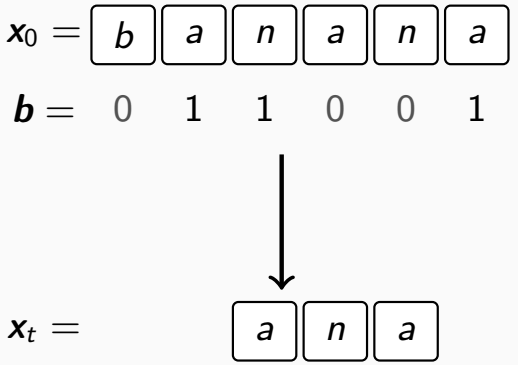




State space: $\mathbb{X} = \bigcup_{n=0}^{\ell_{\max}} \{n\} \times \mathbb{V}^n$, so a state contains both length and tokens. **Forward/noising:** $\boldsymbol{x}_0 \to \boldsymbol{x}_t$, $t \in [0, 1]$, where $\boldsymbol{x}_t$ is a random subsequence of $\boldsymbol{x}_0$ obtained by deleting tokens.

Reverse/generative: $\hat{R}_t^\theta(\boldsymbol{x}, \boldsymbol{y})$ directly parameterizes insertion rates, giving a CTMC that inserts tokens in available gaps. This yields a diffusion-style denoising objective from the CTMC path likelihood. **Key benefit:** sampling quality can trade off with the number of reverse-time steps.

Noising: Random Deletion



The noising process deletes tokens from x_0 to form a subsequence x_t.

1. Sample $t \sim \text{Uniform}[0, T]$.
2. **Joint:** $d \sim \text{Poisson}(\bar{\sigma}_{0,t})$.
3. **Independent:** $d \sim \text{Binomial}\big(n, 1-\exp(-\bar{\sigma}_{0,t})\big)$.
4. Set $d \leftarrow \min(d, n)$.
5. Uniformly at random select d indices from $\{1, \ldots, n\}$ to drop; let $\boldsymbol{b} \in \mathcal{B}_{n,\, n-d}$ enumerate the complementary kept indices.
6. Return $\big((n-d,\ \dot{x}[\boldsymbol{b}]),\ t\big)$.

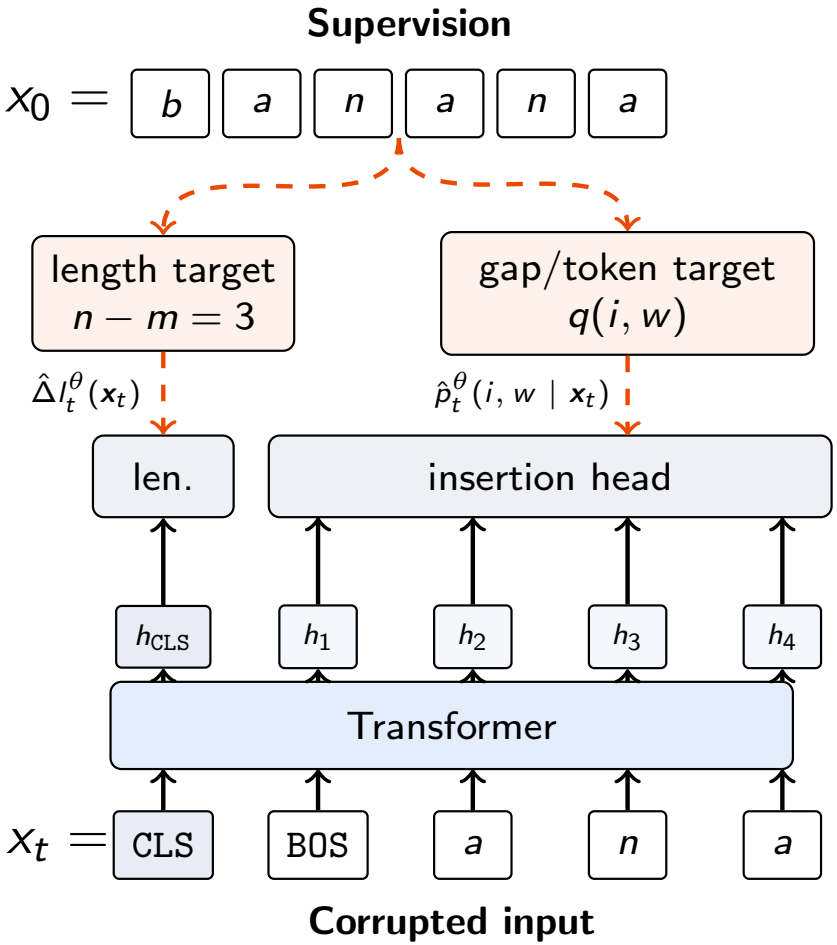
Joint noising

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

Independent noising

$$p^{\text{ind}}_{t|s}\big((m, \dot{y}) \mid (n, \dot{x})\big) = \rho_{s,t}^{m}(1-\rho_{s,t})^{n-m} \sum_{\boldsymbol{b} \in \mathcal{B}_{n,m}} \delta_{\dot{x}[\boldsymbol{b}]}(\dot{y}), \qquad \rho_{s,t} = \exp(-\bar{\sigma}_{s,t}).$$

Denoising: Joint Denoising with single-token Insertion (DILM-S)



DILM-S

Supervision. Delete tokens from x_0 using mask b, giving $x_t = x_0[b]$. The deleted tokens define the length target $n - m$ and the reverse insertion target

$$q(i, w) = \frac{\sum_{k \in s_i(b)} \delta_w(x_0^k)}{n - m}.$$

Predictions. The transformer encodes $\tilde{x}_t = (\text{BOS}, x_t^1, \ldots, x_t^m)$. The length head predicts $\hat{\Delta} l_t^\theta(x_t) = \mathbb{E}[l \mid x_t]$. The insertion head predicts one joint softmax

$$\hat{p}_t^\theta(i, w \mid x_t) \propto \exp s^\theta(i, w \mid x_t).$$

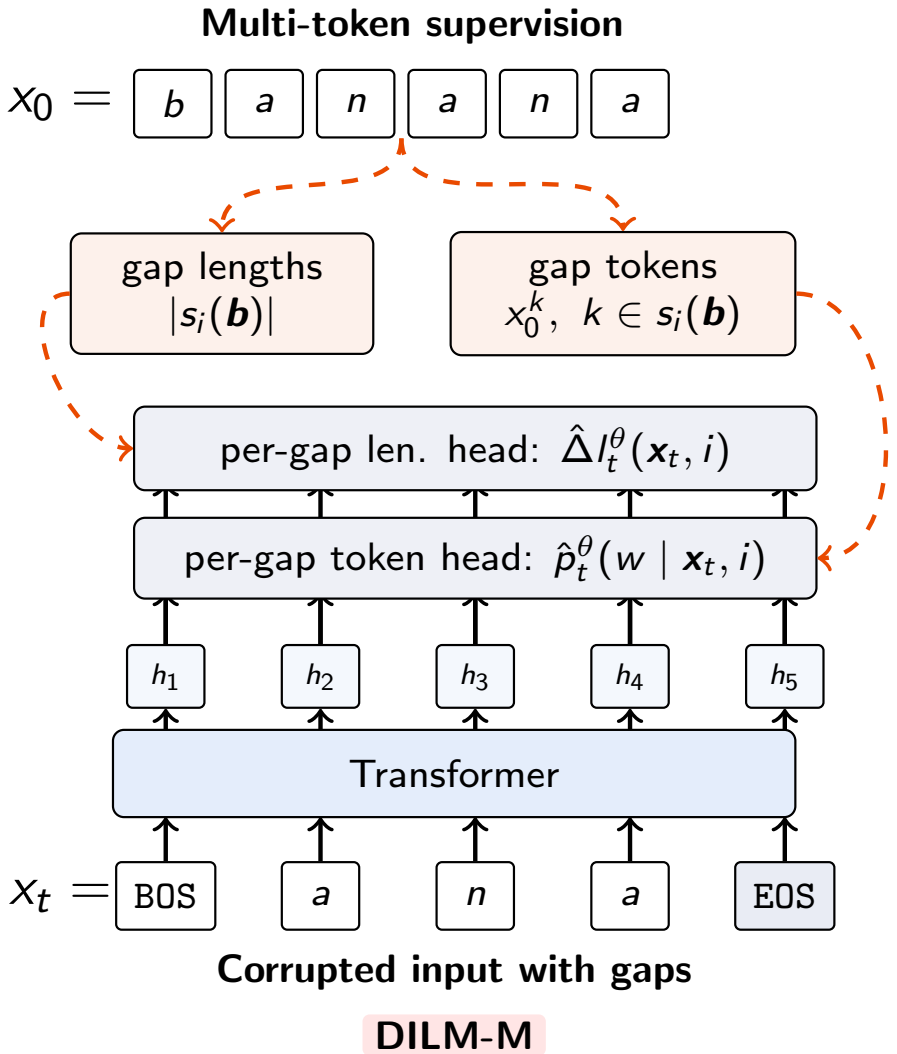
Denoising: Independent Denoising with multi-token Insertion (DILM-M)

Supervision. Independent deletion noise gives a corrupted subsequence $\boldsymbol{x}_t = \boldsymbol{x}_0[\boldsymbol{b}]$. For each gap i, the deleted indices $s_i(\boldsymbol{b})$ define a gap-length target $|s_i(\boldsymbol{b})|$ and token targets

$$x_0^k, \qquad k \in s_i(\boldsymbol{b}).$$

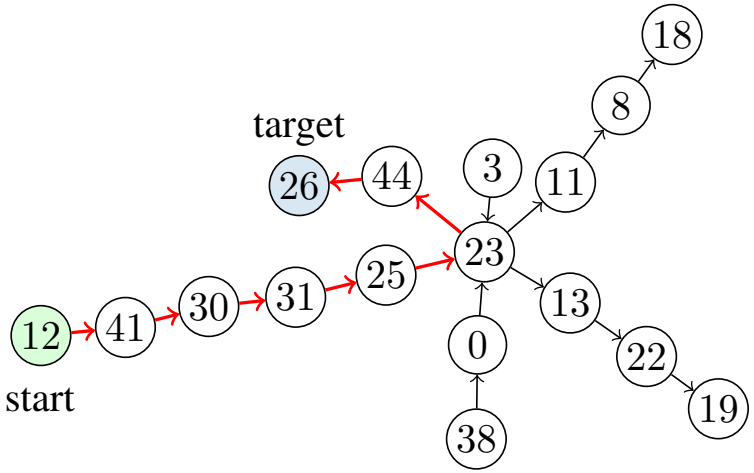
Predictions. The transformer gives one representation per gap. The length head predicts $\hat{\Delta} l_t^\theta(\boldsymbol{x}_t, i) = \mathbb{E}[l \mid \boldsymbol{x}_t, i]$. The token head predicts a separate vocabulary softmax for each gap,

$$\hat{p}_t^\theta(w \mid \boldsymbol{x}_t, i) \propto \exp s^\theta(w \mid \boldsymbol{x}_t, i).$$



Empirical Evaluation: Planning Task

12 41 13 22 ... 11 8 | 12 | 26 | <s> | 12 41 ... 44 26



Given the edges and the start and the target node, the goal is to predict the path from the start to the target node.

Model	Easy	Med.	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	98.6	95.1

ARM: Due to the presence of easy edges, a left-to-right learner like ARM cannot predict the correct edge at the junction

MDM: An arbitrary order learner using absolute positions needs to predict the length of the path to work backwards

DILM: Insertion based models can overcome both these limitations

Empirical Evaluation: Language Modeling

Setup: 12-layer Transformer with 768 hidden dimensions and 12 attention heads, trained from scratch with AdamW on 8 A100 GPUs

LM1B: short news sequences tokenized with the BERT tokenizer and padded to length 128

OpenWebText: longer web-text sequences tokenized with the GPT-2 tokenizer and padded to length 1024

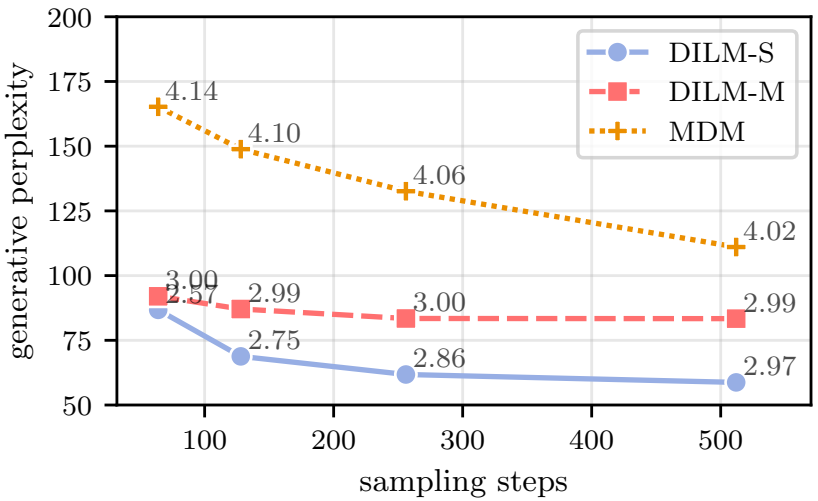
Evaluation: per-token NLL of 1000 unconditional samples under a pre-trained LM

Data	Model	NLL ↓
LM1B	ILM	4.67
	DILM-S	<u>4.65</u>
	DILM-M	4.76
OWT	MDM	3.96
	ILM	4.65
	DILM-S	**3.74**
	DILM-M	<u>3.92</u>

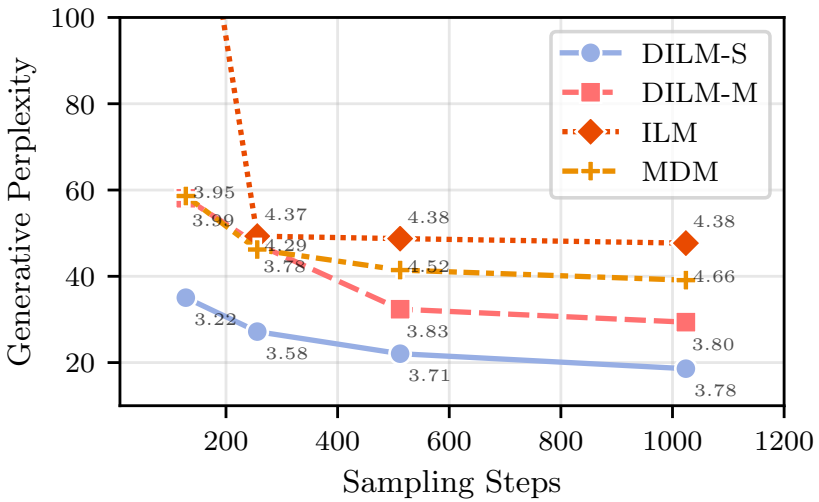
DILM-S and DILM-M improve sample quality as reverse steps increase, giving a smooth quality/-compute trade-off.

Empirical Evaluation: Sampling Trade-off

LM1B



OpenWebText



Generative perplexity improves as the number of sampling steps increases; DILM-S and DILM-M exhibit smoother quality/compute trade-offs than non-diffusion Insertion Language Model (ILM).

Takeaway

CTMC framework: The CTMC framework gives a principled diffusion-style training and sampling methods for insertion language models.
Using this framework, we derive two types of noising and respective generative processes: DILM-S and DILM-M.

Sampling flexibility: We gain sampling flexibility by using the CTMC framework to control the number of sampling steps while keeping the benefits of insertion-based generation.

Future work: Adapting pre-trained autoregressive language models into diffusion-based insertion language models remains an active direction for future work.

Thank you