Logit-KL Flow Matching: Non-Autoregressive Text Generation via Sampling-Hybrid Inference

Egor Sevriugov Nikita Dragunov Anton Razzhigaev Andrey Kuznetsov
Ivan Oseledets

ICLR 2026

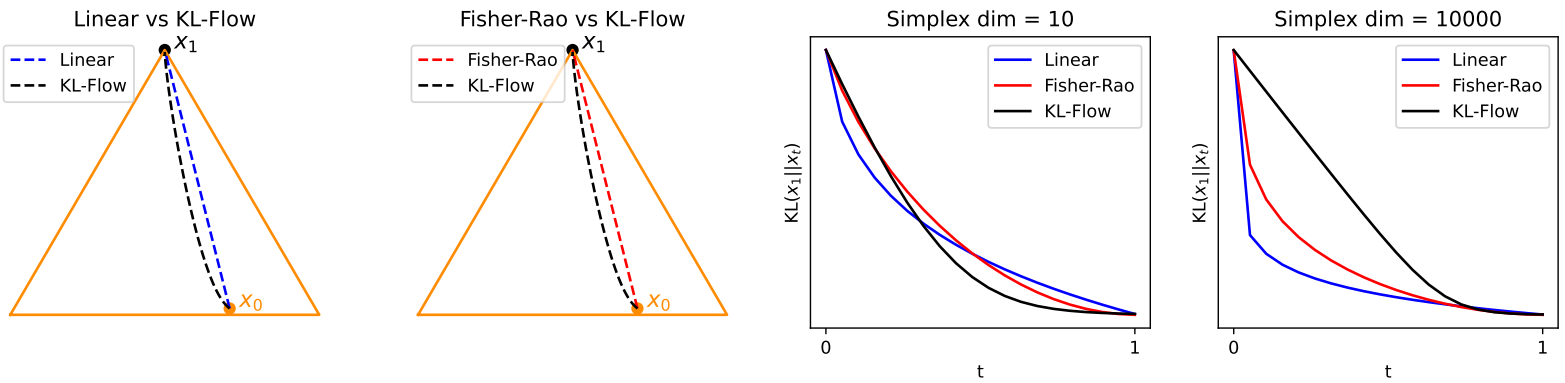
Motivation: fast text generation without quality loss

- Autoregressive decoding is sequential $\Rightarrow$ slow at inference.
- Non-autoregressive (NAR) models generate tokens in parallel $\Rightarrow$ fast.
- Core challenge: capturing dependencies in *discrete* sequences.
- We use **conditional flow matching** (CFM) on the probability simplex.

Goal

Learn a transport from a simple base distribution ρ_0 to text distribution ρ_1, then sample by integrating the learned dynamics.

Key idea 1: KL geodesics (logit-linear paths)



- Standard FM choice: **linear interpolation**, $x_t = (1-t)x_0 + tx_1$.
- We instead use the **KL-geodesic** between $x_0 \sim \rho_0$ and one-hot x_1.
- In logit space this becomes a straight line:

$$l_t = (1-t)\log x_0 + t \log x_1, \qquad x_t = \mathrm{Softmax}(l_t).$$

- For large vocabularies, $\mathrm{KL}(x_1 \| x_t)$ decays more slowly along the KL path, so the denoiser keeps informative gradients longer. This effect is visible in the plots and is aligned with the theory on the next slide.

Under the same setup, geometry alone changes perplexity dramatically.

Geodesic	Llama 2	GPT-3
Linear	1344	15418
Fisher-Rao	192	298
KL-Flow	**41**	**53**

Theoretical guarantees: likelihood recovers the KL-flow

Theorem 1 (single-token recovery)

For sequence length 1, the denoising objective is solved by the posterior expectation of target logits:

$$\hat{v}^{\star}(x_t, t) = \mathbb{E}_{x_1 \sim p(x_1 | x_t)}[\log x_1].$$

Equivalently, learning $p_\theta(x_1 \mid x_t)$ recovers the flow-matching velocity

$$\frac{dl_t}{dt} = \frac{\mathbb{E}_{x_1 \sim p_\theta(x_1 | x_t)}[\log x_1] - l_t}{1 - t}.$$

Theorem 2 (sequence-level tokenwise reduction)

For sequences interpolated tokenwise along the KL-geodesic, the optimal field for token k depends only on the marginal conditional:

$$\hat{v}_\theta^{(k)}(x_t, t) = \mathbb{E}_{x_1^{(k)} \sim p(x_1^{(k)} | x_t)}[\log x_1^{(k)}].$$

So training reduces to minimizing sequence NLL $\mathcal{L} = -\mathbb{E}\left[\sum_{k=1}^{S} \log p_\theta\left(x_1^{(k)} \mid x_t\right)\right]$.

Inference: basic, sampling, and hybrid

Method	Description	Update rule	Limitations
KL-Flow (basic)	Deterministic integration of the learned KL-flow vector field on the simplex.	$\overline{I_1} = \mathbb{E}_{p_\theta(x_1 \mid x_t)}[\, I_1 \,]$ $I_{t+\Delta t} = I_t + \dfrac{\overline{I_1} - I_t}{1 - t}$ $x_{t+\Delta t} = \mathrm{Softmax}(I_{t+\Delta t})$	Higher perplexity, so lower text quality.
KL-Flow (sampling)	Stochastic sampling along the flow using the factorized conditional.	$x_1 \sim p_\theta(x_1 \mid x_t)$ $x_0 \sim p(x_0)$ $x_{t+\Delta t} = \mathrm{interpolate}(x_0, x_1)$	Assumes $p(x_1 \mid x_t) \approx \prod_i p(x_1^{(i)} \mid x_t)$; may reduce diversity.
KL-Flow (hybrid)	Combination of basic and sampling schemes with a switching time $t^\star$.	Basic update for $t \leq t^\star$, sampling update for $t > t^\star$.	Requires tuning $t^\star$.

- **Basic**: high entropy, but very poor quality (perplexity 154.2).
- **Sampling**: best perplexity (3.8), but entropy drops to 1.9, so diversity is low.
- **Hybrid**: keeps entropy high (5.2) while strongly improving quality over basic.
- So, **hybrid gives the best overall trade-off**.

Method	Perplexity ↓	Entropy ↑
KL-Flow (basic)	154.2	5.6
KL-Flow (sampling)	3.8	1.9
KL-Flow (hybrid)	41.4	5.2

Llama-2 perplexity and token-level entropy on FineFineWeb

Results: unconditional, conditional, and code infilling

- **Unconditional LM:** on **FineFineWeb**, KL-Flow (hybrid) beats the strongest prior NAR baseline **SEDD** under the same NFE budget; at 1024 NFE it lowers perplexity from 47.6 to 35.1 on Llama 2, from 74.8 to 54.1 on GPT-3, and from 90.2 to 62.9 on GPT-2. On **TinyStories**, it also improves over **DFM/SEDD** on grammar, creativity, consistency, and Llama 2 perplexity, although GPT-2 still remains strongest there.
- **Conditional generation:** all boosts are **relative to the strongest non-KL baseline in the corresponding metric**. On **Lamini Instruction**, **hybrid** improves Top-5 BLEU by 17%, ROUGE-L by 15%. On **WMT14 De→En**, **sampling** improves Top-5 BLEU by 27%, ROUGE-L by 14%.
- **Code infilling:** on **MBPP** with 10% masking, KL-Flow improves over **DFM**, the best prior baseline here, by 56% on Pass@1 and 14% on Pass@10.

Infilling example from the paper (green = correctly infilled lines)

```
def move_num(test_str):
  res = ''
  dig = ''
  for ele in test_str:
    if ele.isdigit():
      dig += ele
    else:
      res += ele
  res += dig
  return (res)
```

(a) KL-flow

```
def move_num(test_str):
  res = ''
  en = ''
  for ele in test_str:
    if ele.isdigit():
      dig += ele
    else:
      res += ele
  res += dig
  return (res)
```

(b) DFM

```
def move_num(test_str):
  res = ''
  for test_str, ele:
    uid, dig = test_str
    if ele.isdigit():
      dig += ele
    else:
      res += ele
  res += dig
  return (res)
```

(c) GPT 2

```
def move_num(test_str):
  res = ''
  Convert given string
  for ele in (test_str)):
    if ele.isdigit():
      dig += ele
    else:
      res += ele
  res += dig
  return (res)
```

(d) SEDD

Takeaways

What to remember

- **Geometry matters:** KL geodesics $\Leftrightarrow$ logit-linear interpolation gives a much better learning signal.
- **Theory:** conditional likelihood recovers the KL-flow velocity, for both single-token and sequence settings.
- **Practice:** sampling and hybrid inference close the performance gaps of basic ODE integration.

More details could be found in the paper; thanks for watching!