ONE-STEP TEXT GENERATION BY SEQ-DRIFTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models generate tokens in parallel, but still require multiple sampling steps to produce high-quality text. In this paper, we investigate whether a model can learn to generate an entire sequence in a single forward pass. We propose *Seq-Drifting*, a framework that applies drifting to sequences of continuous token representations and directly learns a map from Gaussian noise to text. Unlike existing DLMs that derive the supervision at each position directly from the observed token in a training sequence, Seq-Drifting builds a drift field whose position-wise targets adapt to the current generated context. At each position, attraction guides each output toward a set of contextually plausible token representations, while repulsion discourages collapse across samples and repetition within sequences. The generator learns from this drift field during training and produces all tokens jointly in one pass at inference. Empirically, Seq-Drifting achieves a generative perplexity of 36.01 on LM1B in one step and performs competitively in both unconditional and conditional generation. Experiments on translation, summarization, and reasoning further demonstrate the applicability of the framework beyond open-ended generation. These results suggest that drifting over token sequences offers a promising path toward effective one-step text generation.

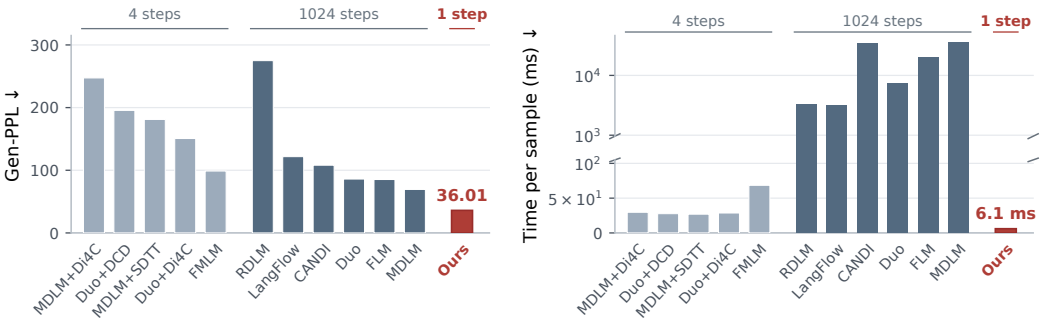

Figure 1: **Seq-Drifting** achieves lower generative perplexity than prior few-step distilled DLMs, and is on par with or better than multi-step DLMs, with a 4–11× speedup over the few-step baselines and a 900–2500× speedup over the multi-step ones in wall-clock time. (Gen-PPL under GPT-2 Large; dataset: LM1B, T=128. Detailed comparison in Figure 3 and Table 6.)

1 INTRODUCTION

Autoregressive (AR) language models are the dominant paradigm for text generation and have shown strong capabilities across a wide range of domains (Radford et al., 2019; Brown et al., 2020). At inference, however, they generate one token at a time, making sequential decoding a fundamental latency bottleneck that grows with sequence length. Non-autoregressive alternatives such as diffusion language models (DLM) generate multiple tokens in parallel (Sahoo et al., 2024; Li et al., 2022; Hu et al., 2026; Lee et al., 2026), but they still rely on multiple refinement steps to produce high-quality samples. Distillation methods such as SDTT (Deschenaux & Gulcehre, 2025), Di4C (Hayakawa et al., 2025), and FMLM (Lee et al., 2026) reduce the number of steps, but generation quality remains limited at small sampling budgets. A key question, then, is whether we can directly learn high-quality one-step generation through a single forward pass from noise to text.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Most generative models can be seen as a map from a noise vector to data. Because this map is highly multimodal, previous methods often need to decompose it into many small steps. Drifting (Deng et al., 2026) offers a different route, learning the map directly as a single forward pass in continuous domains. Rather than learning a trajectory that is unrolled at inference, a drifting model evolves its pushforward map during training. As the parameters are updated, each noise sample is carried a little closer toward the data distribution, a movement governed by a drifting field that vanishes once the generated and data distributions match. At convergence the map can reach a fixed point, so a single forward pass can turn noise into a sample without an iterative sampler at inference.

Although drifting has shown promising results on images, transferring it to text is far from straightforward. A text sample is a sequence in which every position contains a token drawn from a finite vocabulary. Since valid text sequences are composed of discrete token choices, a continuous drift step will generally land between valid token embeddings rather than on one. The drift may therefore operate in regions containing no valid token. The space is also semantically non-smooth, as a single substituted token can overturn the meaning of a sentence or make it incoherent, which means nearness in embedding space does not necessarily imply nearness in meaning. One might hope to sidestep these issues by drifting a single sentence-level embedding, where the space is continuous. However, this only relocates the difficulty, because mapping such a vector back to a faithful token sequence is itself a hard sequence-decoding problem, and a single vector exposes none of the per-position targets a drift step needs.

In this work we introduce **Seq-Drifting**, which adapts drifting to sequences of continuous token representations and learns the noise-to-text map in a single forward pass. Our core idea is to provide each position with supervision from a context-dependent set of plausible tokens. This support set anchors the drift to actual vocabulary tokens and identifies which choices are compatible with the current generated context. Seq-Drifting then attracts each output toward these tokens, allowing different noise inputs to select different valid alternatives without pulling them all toward an average representation. Because the support tokens depend on the generated context, the supervision couples token choices across positions and guides the generator toward coherent sequences.

Seq-Drifting establishes a new framework for one-step text generation and brings three advantages. First, it generates a sample in a single forward pass, so its inference cost does not grow with a step budget as in multi-step diffusion and few-step distillation, while matching their quality at a fraction of the inference cost. Second, it addresses the ambiguity of one-step generation without relying on iterative refinement. Rather than assigning each noise input a unique target sequence, it allows different noise to naturally settle on their own coherent continuations while preserving multiple generation modes. Finally, it unifies two ways of learning the one-step map by constructing support sets either from corpus retrieval or scoring by a frozen language model, which opens the possibility of turning an existing autoregressive language model into a one-step generator.

Empirically, we show that Seq-Drifting combines competitive generation quality with substantially lower inference cost. For example, in Figure 1, it reaches a generative perplexity of 36.01 with an $11.2\times$ wall-clock speedup over the strongest four-step distilled baseline, even outperforming several DLMs that use 1024 sampling steps. This advantage extends to prefix-conditioned generation, such as machine translation, summarization, and reasoning, which broadens its applicability beyond language continuation. Overall, these results demonstrate the feasibility of directly learning a one-step text generator and point to a new direction for diffusion language models.

2 Related Work

Diffusion language models (DLMs) provide an iterative non-autoregressive alternative to the token-by-token decoding of autoregressive models (Radford et al., 2019; Brown et al., 2020), and can be broadly categorized into discrete and continuous formulations (Li et al., 2025).

Discrete DLMs operate directly over categorical tokens. D3PM (Austin et al., 2021) provides a general framework for discrete corruption, while MDLM (Sahoo et al., 2024) uses masking to learn token recovery from partially observed sequences. LLaDA (Nie et al., 2025) further demonstrates masked diffusion at the scale of large language models. Unlike standard irreversible unmasking, the uniform-state formulation of Duo (Sahoo et al., 2025) allows previously generated tokens to be revised. Block diffusion methods (Arriola et al., 2025; 2026) instead combine parallel generation

within blocks with autoregressive conditioning across blocks. However, parallel per-position predictions do not by themselves capture the joint uncertainty of tokens updated simultaneously, making inter-token dependencies particularly important when sampling is compressed into large jumps.

Continuous DLMs instead model language generation in a continuous representation space. Diffusion-LM (Li et al., 2022) and DiffuSeq (Gong et al., 2023) operate on token embeddings, whereas SSD-LM (Han et al., 2023) and TESS (Mahabadi et al., 2024) use simplex-based representations. Beyond the state space, an important distinction is how discrete supervision constrains the trajectory. FLM (Lee et al., 2026) learns continuous denoising over one-hot encodings with token-level cross-entropy, and LangFlow (Chen et al., 2026) connects embedding-space diffusion to Flow Matching through Bregman divergence. ELF (Hu et al., 2026) instead reserves token-level supervision for the final decoding step, allowing the preceding dynamics to remain in continuous embedding space. These advances improve continuous generation, but an effective trajectory still requires tens or even hundreds of sampling steps at inference.

Few-step methods address this problem by learning to traverse or replace an existing sampling trajectory. SDTT (Deschenaux & Gulcehre, 2025) compresses denoising through self-distillation, Di4C (Hayakawa et al., 2025) explicitly addresses the dimensional correlations lost in factorized reverse transitions, and FMLM (Lee et al., 2026) distills a continuous flow into a map between distant timepoints. Drifting (Deng et al., 2026) provides a different training formulation, evolving the generator's outputs through attraction and repulsion while retaining a single forward pass at inference. Seq-Drifting extends this formulation to token sequences. Context-dependent, per-position supervision supplies plausible token targets for its current outputs. The support sets can draw on corpus retrieval or a frozen language model, while the objective directly trains the one-step map without reproducing a diffusion or flow trajectory.

3 PRELIMINARIES

We consider a one-step generator G_θ that maps a noise input $\epsilon \sim p_\epsilon$ to an output $\boldsymbol{x} = G_\theta(\epsilon)$ with distribution q_θ, and training seeks $q_\theta \approx p_{\mathrm{data}}$. Since the parameters are updated step by step, the same noise ϵ produces a slightly different output after each step. The displacement

$$\Delta \boldsymbol{x}_i \;=\; G_{\theta_{i+1}}(\epsilon) - G_{\theta_i}(\epsilon) \tag{1}$$

comes entirely from the parameter update, and is called the *drift* of $\boldsymbol{x}$ at iteration i. A *drifting field* $V_{p,q}$ (Deng et al., 2026) prescribes this drift sample by sample, with p fixed to the target p_{data} and q to the current model q_{θ_i},

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + V_{p_{\mathrm{data}},\, q_{\theta_i}}(\boldsymbol{x}_i). \tag{2}$$

The field vanishes when the two distributions agree, $V_{p,p} = \mathbf{0}$, so $q_\theta = p_{\mathrm{data}}$ is a fixed point of training. Regressing each output onto its drifted target with a stop gradient turns this into the loss,

$$\mathcal{L}(\theta) = \mathbb{E}_\epsilon \big\| \, G_\theta(\epsilon) - \mathrm{sg}\big(G_\theta(\epsilon) + V(G_\theta(\epsilon))\big) \big\|^2, \tag{3}$$

whose value equals $\mathbb{E}_\epsilon \|V(G_\theta(\epsilon))\|^2$, so minimizing it drives the drift toward zero.

A concrete field comes from a kernel mean-shift. For a sample $\boldsymbol{x}$, a positive set $\mathcal{P}$ drawn from p_{data} attracts and a negative set $\mathcal{N}$ drawn from q_θ repels,

$$V(\boldsymbol{x}) = \underbrace{\sum_{\boldsymbol{y}\in\mathcal{P}} \tilde{k}(\boldsymbol{x},\boldsymbol{y})\,(\boldsymbol{y}-\boldsymbol{x})}_{\text{attraction}} \;-\; \underbrace{\sum_{\boldsymbol{y}\in\mathcal{N}} \tilde{k}(\boldsymbol{x},\boldsymbol{y})\,(\boldsymbol{y}-\boldsymbol{x})}_{\text{repulsion}}, \tag{4}$$

where $\tilde{k}$ is a similarity kernel normalized over each set. The attraction moves $\boldsymbol{x}$ toward the data it already resembles and the repulsion moves it off the other samples, and the two cancel once the generated and data distributions agree.

Defining the drift target. The field in Eq. (4) specifies how a generated sample should move during training. Positive samples provide the destinations for attraction, while other generated samples provide the reference points for repulsion. The similarity kernel determines which samples contribute most strongly to each force. Combining the forces gives a displacement, which defines the stop-gradient target $\boldsymbol{x} + V(\boldsymbol{x})$ in Eq. (3). The generator learns to approach this target through parameter updates. Adapting drifting to a new domain therefore requires suitable positive and negative samples, a geometry for comparing them, and a rule for converting the field into a training target.

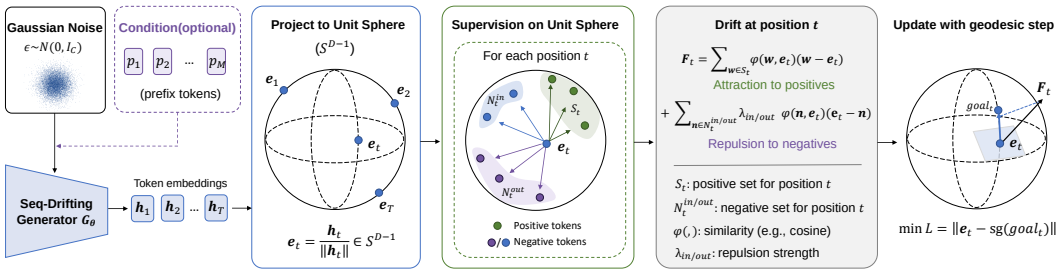

Figure 2: **Seq-Drifting.** In one forward pass the generator G_θ maps a noise vector $\epsilon \sim \mathcal{N}(0, I_C)$ (optionally with a prefix $p_1, \ldots, p_M$) to token embeddings $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T$, each projected onto the unit sphere as $\boldsymbol{e}_t = \boldsymbol{h}_t / \|\boldsymbol{h}_t\| \in S^{D-1}$. Supervision is applied *per position*. A positive set $\mathcal{S}_t$ of plausible tokens attracts $\boldsymbol{e}_t$, while the negative sets N_t^{out} (other samples) and N_t^{in} (other positions of the same sample) repel it, weighted by a similarity kernel φ and strengths $\lambda_{\text{in/out}}$. The two combine into the drift force $\boldsymbol{F}_t$; a single geodesic step along $\boldsymbol{F}_t$ gives the stop-gradient target $goal_t$, onto which the generator is regressed, $\mathcal{L} = \mathbb{E}\|\boldsymbol{e}_t - \mathrm{sg}(goal_t)\|^2$.

Difficulties in applying drifting to text. A text sequence is composed of discrete token choices, which creates three difficulties for drifting. First, valid outputs lie in a discrete and semantically non-smooth space. A continuous displacement can change the decoded token, but proximity between embeddings does not guarantee that the new token fits the surrounding sequence. Second, the tokens that form a coherent continuation depend on the preceding sequence. However, a training example supplies only the continuation observed after its own context, which may differ substantially from the generator's current context. Using that observed continuation directly as a drift target can therefore yield a weak or misaligned update. Third, under cosine decoding, radial separation can increase Euclidean distances without changing token identity. Unconstrained repulsion can therefore spend its force on growing norms while producing no additional token-level diversity.

4 Seq-Drifting

To address the difficulties above, Seq-Drifting represents a sequence with per-position token representations and provides each position with a context-dependent set of plausible tokens. These representations and targets are placed in a shared spherical space, where the resulting drift remains anchored to valid vocabulary while the noise-to-text map is learned directly. Figure 2 summarizes the framework. We first describe the one-step generator and its representation space, then introduce support-set construction, the attractive and repulsive forces, and the training objective.

4.1 Spherical one-step generator

We generate a sequence of continuous representations, each one corresponding to a token. This allows the generator to produce all positions jointly while exposing each token choice to local drift supervision. Specifically, our generator G_θ maps a shared noise vector $\epsilon \in \mathbb{R}^C$ to T token representations $\{\boldsymbol{h}_t\}_{t=1}^T$, with $\boldsymbol{h}_t \in \mathbb{R}^D$, in one forward pass. Architectural details are given in Appendix A.1. Then, we recover tokens by cosine similarity to a frozen vocabulary embedding table. Since this decoding rule depends only on direction, we normalize both the output vectors $\boldsymbol{h}_t$ and the vocabulary embeddings $\boldsymbol{w}_v$, i.e., $\boldsymbol{e}_t = \boldsymbol{h}_t / \|\boldsymbol{h}_t\| \in S^{D-1}$ and $\|\boldsymbol{w}_v\| = 1$. Attraction, repulsion, and token recovery then depend on the same directional geometry. The decoded token is $\mathrm{tok}_t = \arg\max_v \boldsymbol{e}_t^\top \boldsymbol{w}_v$, with non-text tokens excluded. This operation produces the final text at inference, requiring no separate vocabulary softmax.

4.2 Context-dependent token support

The main challenge in selecting positives is to provide useful guidance for the sequence the generator currently produces. A reference sentence may differ from that sequence at many positions, so its tokens need not be suitable targets for the corresponding positions. We instead construct positive targets at each position, asking which tokens are plausible given the generated context.

For position t, we collect the positive targets into a support set $\mathcal{S}_t$ (i.e., $\mathcal{P}$ in § 3), with each candidate represented by its unit embedding. The set supplies concrete token targets without prescribing a single continuation. It separates contextual plausibility from the choice made by an individual sample. The support set determines which tokens are eligible for attraction, while the local kernel introduced next determines how strongly each candidate attracts the current output.

There are two natural ways to form $\mathcal{S}_t$. The *off-policy* route reads it from the corpus, matching each generation against real sequences and taking the tokens they place at position t, so the candidates come from observed continuations. The *on-policy* route queries a frozen language model (hereafter, the *support model*) on the generator's current prefixes to propose contextually plausible next-token candidates, where one forward pass over the generated sequence returns a next-token distribution p_t at every position at once. $\mathcal{S}_t$ collects its most probable tokens (the top k), so the target adapts to wherever the current generator sits. The same construction also applies to a closed-source support model, which can be queried multiple times to sample continuations from the generated context and form $\mathcal{S}_t$. We empirically find the on-policy support the stronger of the two (§5.4) and adopt it by default. The support model is queried only during training to construct the support set $\mathcal{S}_t$, and is not required at inference. The resulting supervision is set-valued and context-dependent rather than a fixed per-position target.

4.3 Angular mode-seeking attraction

The attraction pulls each position toward the supported token nearest its own embedding. We measure closeness by angular distance $d(\boldsymbol{e}_t, \boldsymbol{w}) = 1 - \boldsymbol{e}_t^\top \boldsymbol{w}$ and form a low-temperature similarity kernel φ, normalized over the support set,

$$\varphi(\boldsymbol{w}, \boldsymbol{e}_t) \;\propto\; \exp\Big(-\frac{d(\boldsymbol{e}_t, \boldsymbol{w})}{s_t\,\tau}\Big), \qquad \boldsymbol{u}_t = \sum_{\boldsymbol{w}\in\mathcal{S}_t} \varphi(\boldsymbol{w}, \boldsymbol{e}_t)\,(\boldsymbol{w} - \boldsymbol{e}_t), \qquad s_t = \mathrm{std}_{\boldsymbol{w}\in\mathcal{S}_t}\, d(\boldsymbol{e}_t, \boldsymbol{w}), \tag{5}$$

where s_t is the standard deviation of $d(\boldsymbol{e}_t, \boldsymbol{w})$ for candidates at position t and τ is a temperature. Normalizing by s_t accounts for each position's candidate spread, while $\tau < 1$ sharpens the kernel around nearby candidates. It is therefore mode-seeking and concentrates the weight on the near supported token, so different noise inputs commit to different tokens rather than to the blurred centroid of $\mathcal{S}_t$. The mean-shift then gives the attractive force $\boldsymbol{u}_t = \sum_{\boldsymbol{w}\in\mathcal{S}_t} \varphi(\boldsymbol{w}, \boldsymbol{e}_t)(\boldsymbol{w} - \boldsymbol{e}_t)$.

4.4 Per-position angular repulsion

The repulsion stops distinct noise inputs from collapsing to the same output. It acts per position against two negative sets, i.e., $\mathcal{N} \triangleq \mathcal{N}_t = \{N_t^{\mathrm{out}}, N_t^{\mathrm{in}}\}$. The cross-sample set N_t^{out} contains the position-t representations from the other samples in the batch. The intra-sequence set N_t^{in} contains the representations at the other positions of the same sequence, and pushing away from it suppresses within-sentence repetition. Each uses the same angular kernel φ as the attraction,

$$\boldsymbol{r}_t^{\mathrm{out}} = \sum_{\boldsymbol{n}\in N_t^{\mathrm{out}}} \varphi(\boldsymbol{n}, \boldsymbol{e}_t)\,(\boldsymbol{e}_t - \boldsymbol{n}), \qquad \boldsymbol{r}_t^{\mathrm{in}} = \sum_{\boldsymbol{n}\in N_t^{\mathrm{in}}} \varphi(\boldsymbol{n}, \boldsymbol{e}_t)\,(\boldsymbol{e}_t - \boldsymbol{n}). \tag{6}$$

Both are then rescaled to the magnitude of the attraction before the update weights them by λ_{out} and λ_{in}. Restricting the cross-sample set N_t^{out} to the samples that share a context is studied in §5.4.

On the sphere, repulsion separates outputs by changing their directions rather than increasing their norms. This prevents radial growth and aligns repulsion with cosine decoding. Together with the bounded geodesic update in §4.5, it defines the displacement of each training target.

4.5 Geodesic update and training

The update sums the attraction and the two repulsions into one drift force, $\boldsymbol{F}_t = \boldsymbol{u}_t + \lambda_{\mathrm{out}}\,\boldsymbol{r}_t^{\mathrm{out}} + \lambda_{\mathrm{in}}\,\boldsymbol{r}_t^{\mathrm{in}}$, projects it onto the tangent plane at $\boldsymbol{e}_t$ to drop the radial part, $\boldsymbol{F}_t' = \boldsymbol{F}_t - (\boldsymbol{F}_t^\top \boldsymbol{e}_t)\,\boldsymbol{e}_t$, and applies it as one step along the great circle by the exponential map,

$$goal_t = \cos(\alpha)\,\boldsymbol{e}_t + \sin(\alpha)\,\frac{\boldsymbol{F}_t'}{\|\boldsymbol{F}_t'\|}, \qquad \alpha = \min\big(\|\boldsymbol{F}_t'\|,\ \alpha_{\max}\big). \tag{7}$$

Here we use the norm of $\boldsymbol{F}'_t$ as the arc length of the geodesic step, capped by $\alpha_{\max}$ to prevent overshooting. Treating $goal_t$ as a fixed target during differentiation, the loss of Eq. (3) becomes the squared distance $\mathbb{E}\,\|\boldsymbol{e}_t - \mathrm{sg}(goal_t)\|^2$, which is bounded because both points lie on the sphere. Appendix A.2 analyzes the conditions under which the field vanishes as the temperature approaches zero. Appendix A.3 provides the training pseudocode for the on-policy support-set implementation.

Conditional generation. For conditional generation, we additionally provide the generator with the conditioning token embeddings, which the output positions access through cross-attention. The support model also receives the conditioning sequence followed by the generated continuation to provide token support sets. To stabilize learning this conditional map from scratch, we first train with an unconditional warm-up. During this phase, the context is filled with a constant padding token (e.g., `<|endoftext|>` for GPT-2) under normal attention, which keeps the conditioning path active while the generator learns fluent text. We then replace the constant context with the real prefix, allowing the generator to learn context dependence from this fluent initialization.

5 Experiments

5.1 Setup

Baselines. Our main comparison is against the *few-step distilled models* that compress a multi-step sampler into a handful of steps, including FMLM (Lee et al., 2026), Duo+DCD (Sahoo et al., 2025), MDLM+SDTT (Deschenaux & Gulcehre, 2025), and Duo/MDLM+Di4C (Hayakawa et al., 2025). We report them at 1, 2, and 4 steps. We also compare against *multi-step models* that generate through multiple (e.g., 1024) denoising steps, namely FLM (Lee et al., 2026), Duo (Sahoo et al., 2025), MDLM (Sahoo et al., 2024), CANDI (Pynadath et al., 2026), RDLM (Jo & Hwang, 2026), LangFlow (Chen et al., 2026), and ELF (Hu et al., 2026).

Data and protocol. We follow the evaluation protocol of Lee et al. (2026) and report at two output lengths. In the length-128 setup the models are trained on One Billion Word (LM1B) (Chelba et al., 2013), and in the length-1024 setup on OpenWebText (Gokaslan et al., 2019) using the corresponding benchmark preprocessing. We draw 1024 samples per configuration. For each method we report the number of forward passes per sample (**Step**), the generative perplexity of the samples under GPT-2 Large (**Gen-PPL**↓), and their unigram entropy (**Entropy**).

Implementation details. We consider two implementations of on-policy support-set construction. For **Seq-Drifting**^{S}, we first train a GPT-2 from *scratch* on the same corpus as the baselines for 10^5 steps, then freeze it and use it as the support model. This setting uses no additional text beyond the baselines. For **Seq-Drifting**^{P}, we directly use a *public* pretrained GPT-2 checkpoint to construct the support sets in the same way. The support set uses $k = 64$, the attraction temperature is $\tau = 0.3$, and the cross-sample and intra-sequence repulsions use $\lambda_{\text{out}} = 5$ and $\lambda_{\text{in}} = 100$. The geodesic step is capped at $\alpha_{\max} = 1.5708$ radians. We train with AdamW (learning rate 2×10^{-4}, weight decay 0, cosine schedule) on 4 H200 GPUs. Unconditional models are trained for 2×10^5 steps. Conditional models use a 2×10^5-step unconditional warm-up, followed by 1×10^5 conditional steps, keeping an exponential moving average of the weights with decay 0.999. Therefore, the total training step of Seq-Drifting is shorter than the 1.0–1.5M-step recipes usually used by the baselines. The effective batch is 2048 at length 128 and 128 at length 1024, the longer sequence being reduced to fit memory.

5.2 Unconditional generation

Figure 3 compares the few-step distilled models with Seq-Drifting at a single pass. The full multi-step results are reported in Table 6 in Appendix B.3. Seq-Drifting^S produces competitive unconditional samples. On LM1B, it outperforms the strongest four-step baseline and even beat several 1024-step baselines, which directly reflect the effectiveness of our drifting framework. Seq-Drifting^P further reduces Gen-PPL relative to FMLM by 69.8% on LM1B and 24.2% on OpenWebText. The entropy and Self-BLEU results in Appendix B.1 show that they preserve generation diversity. To verify that the comparison is not tied to the GPT-2 Large evaluator, Appendix B.4 recomputes Gen-PPL with Pythia-1B (Biderman et al., 2023) and OPT-1.3B (Zhang et al., 2022). The resulting comparisons remain consistent with the main table, showing that

Method	Step	LM1B		OWT	
		Gen-PPL↓	Ent.	Gen-PPL↓	Ent.
FMLM	1	119.34	4.16	169.32	5.17
	2	110.19	4.21	135.68	5.30
	4	98.76	4.21	111.49	5.36
Duo+DCD	1	1006.75	4.33	6330.89	5.98
	2	466.46	4.24	983.80	5.52
	4	195.52	4.23	252.04	5.49
Duo+Di4C	1	292.94	3.79	370.51	3.92
	2	247.69	3.87	210.22	4.63
	4	150.67	4.00	154.67	4.85
MDLM+SDTT	1	1099.64	4.23	1264.02	5.25
	2	448.78	4.22	860.40	5.34
	4	181.08	4.23	319.73	5.36
MDLM+Di4C	1	1217.10	4.38	1272.27	5.28
	2	621.59	4.37	742.99	5.34
	4	247.32	4.00	226.95	5.37
Seq-Drifting^S	**1**	**93.81**	**4.21**	**183.98**	**4.95**
Seq-Drifting^P	**1**	**36.01**	**4.42**	**128.33**	**5.08**

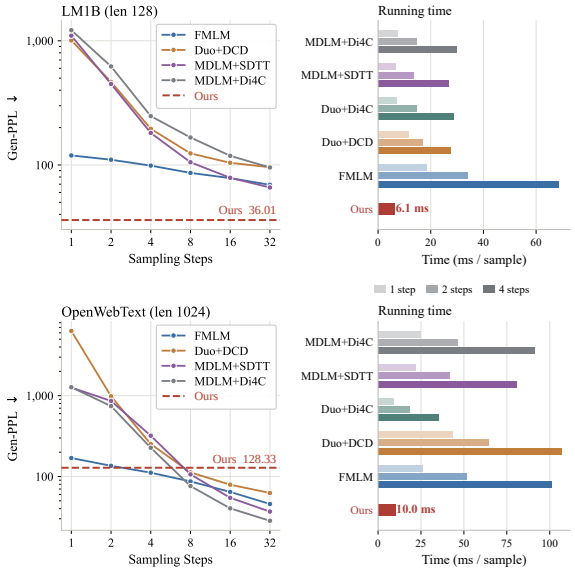

Figure 3: **Unconditional generation.** *Left:* Gen-PPL (under GPT-2 Large) and unigram entropy for the few-step distilled models at 1/2/4 steps and Seq-Drifting at a single pass, on LM1B (T=128) and OpenWebText (T=1024); multi-step baselines are in Table 6. *Right:* Gen-PPL vs. sampling steps and wall-clock running time per sample, showing Seq-Drifting achieves lower Gen-PPL in a single forward pass at a fraction of the cost.

the observed gains are not specific to the evaluator. The running-time in Figure 3(right) shows that Seq-Drifting has the lowest single-sample latency compared with the baselines. Relative to four-step FMLM, which reaches a comparable quality level, Seq-Drifting is 11.2× faster on LM1B and 10.1× faster on OpenWebText. A broader comparison of sampling time across multi-step baselines is provided in Appendix B.2. Qualitative samples are shown in Appendix B.9.

5.3 Conditional Generation

Table 1 summarizes two conditional-generation settings, each with a 64-token context. On LM1B the model generates 64 output tokens. On OpenWebText it generates 512 output tokens so that the context and output remain within GPT-2's 1024-token context window. Both Seq-Drifting^S and Seq-Drifting^P outperform all few-step distilled baselines in both settings. Compared with unconditional generation, the gap between Seq-Drifting^S and Seq-Drifting^P narrows, and Seq-Drifting^S is even stronger on the 512-token output setting. One possible explanation is that the observed prefix narrows the plausible continuations, reducing ambiguity in the support sets and making the conditional map easier to learn. Overall, the results suggest that Seq-Drifting can preserve both fluency and conditional relevance in conditional generation.

Table 1: **Conditional generation.** A 64-token prefix followed by a continuation.

Method	Step	64→64		64→512	
		Gen-PPL↓	Ent.	Gen-PPL↓	Ent.
FMLM	1	162.20	3.78	418.37	3.32
	2	143.62	3.82	441.09	3.55
	4	124.74	3.82	320.22	3.52
Duo+DCD	1	972.09	3.77	7639.94	3.85
	2	384.35	3.72	846.20	3.72
	4	209.32	3.74	205.79	3.75
Duo+Di4C	1	409.44	3.02	2784.97	3.74
	2	507.34	3.30	1007.87	3.75
	4	476.22	3.48	254.55	3.75
MDLM+SDTT	1	912.05	3.75	2770.06	3.74
	2	333.40	3.74	1077.27	3.75
	4	139.73	3.74	298.50	3.75
MDLM+Di4C	1	1018.23	3.83	2791.44	3.74
	2	509.84	3.83	944.53	3.75
	4	231.36	3.82	226.34	3.75
Seq-Drifting^S	**1**	**79.13**	**3.74**	**119.13**	**4.10**
Seq-Drifting^P	**1**	**76.16**	**4.40**	**168.34**	**4.78**

5.4 Ablations

We report ablations in both the unconditional (128) and conditional (64→64) settings in Table 2. The geometry ablation shows that the spherical update is essential. Replacing it with a Euclidean update allows repulsion to increase the embedding norm and causes training to diverge. The positive-

Table 2: **Ablation study** on unconditional (128) and conditional (64→64) generation.

Decision	Variant	Unconditional (128)		Conditional (64→64)	
		Gen-PPL↓	Entropy	Gen-PPL↓	Entropy
Seq-Drifting^P		36.01	4.42	76.16	4.40
Geometry	Euclidean	*diverges (loss blow-up)*			
Positives	off-policy	>1000	>6	>10000	>4
Anti-repetition	w/o intra	10.01	1.82	12.18	1.31
Repulsion	per-context (block)	–	–	81.27	4.27
Warm-up	w/o warm-up	–	–	53.57	3.87

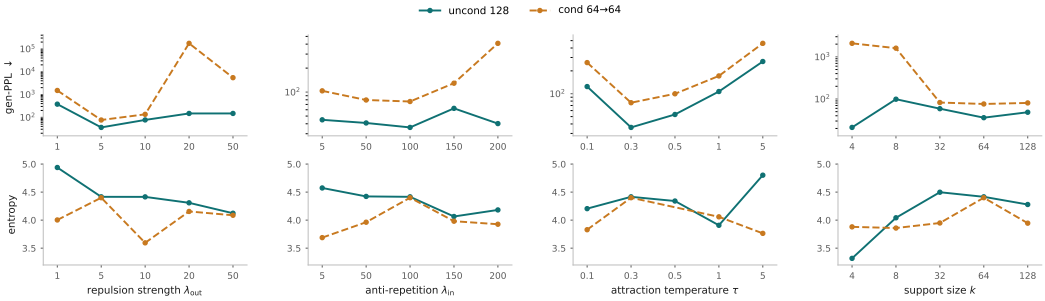

Figure 4: **Hyperparameter sweeps** of the repulsion strength λ_{out}, the anti-repetition weight λ_{in}, the attraction temperature τ, and the support size k. Setting: unconditional 128-token generation (solid), conditional 64→64 generation (dashed).

selection ablation highlights that support construction should match the task. For open-ended generation, fixed corpus-based targets do not follow the generator's evolving contexts. When the attraction becomes sharp, this mismatch can produce oscillatory training and rising generative perplexity and entropy (Appendix B.8). For tasks with explicit target answers, dataset-derived supervision can remain appropriate, as in our reasoning experiments in §5.6. For repulsion, the cross-sample term is not enough on its own. An intra-sequence term that pushes each position off the other positions of the same generation is also necessary, and dropping it (*w/o intra*) collapses the output into within-sentence token repetition. In the conditional setting, whether the cross-sample repulsion is restricted to samples that share a prefix makes little difference. Finally, unconditional warm-up stabilizes from-scratch conditional training by allowing the generator to learn fluency before it must also model prefix dependence.

5.5 Hyperparameters

Figure 4 further sweeps parameters $\lambda_{\text{out}}, \lambda_{\text{in}}, \tau$, and k. We observe that the first three parameters show the same broad pattern. Gen-PPL first improves and then worsens as the parameter moves away from a favorable middle range, revealing a balance between a target that is too weak and one that is too sharp or too disruptive.

λ_{out} has a larger effect than λ_{in}. When repulsion between different noise samples is too weak, the generator is encouraged to learn an average token direction rather than preserve distinct modes. λ_{in} mainly controls repetition within a sequence and therefore provides a narrower correction. For τ, making the temperature too small makes the target at each position nearly unique, discarding the useful ambiguity in the support tokens. For the support size, $k = 4$ collapses the output (entropy below 3.5). Increasing the support size generally restores diversity and improves Gen-PPL.

5.6 Downstream task evaluation

Translation and summarization. Following Hu et al. (2026), we evaluate Seq-Drifting on WMT14 (German–English translation) (Bojar et al., 2014) and XSum (abstractive summarization) (Narayan et al., 2018). Experimental details are provided in Appendix B.5. Table 3 shows that Seq-Drifting outperforms the few-step baselines on translation and on most summarization metrics,

Model	Step	WMT14	XSum		
		BLEU↑	ROUGE-1↑	ROUGE-2↑	ROUGE-L↑
AR	–	25.2†	30.5±0.13†	10.2±0.11†	24.4±0.12†
MDLM	4	< 10	19.0±0.06	2.4±0.01	13.2±0.04
	64	18.4†	33.4±0.11†	11.6±0.10†	25.8±0.10†
Duo	4	< 10	24.4±0.06	5.6±0.01	18.1±0.04
	1000	21.3†	31.4±0.12†	10.1±0.10†	25.0±0.12†
ELF	1	11.5	23.7±0.11	5.7±0.08	19.4±0.10
	64	26.4†	36.0±0.13†	12.2±0.11†	27.8±0.12†
Seq-Drifting (vs. dataset)	1	18.6	24.7±0.36	5.3±0.24	20.1±0.32
Seq-Drifting (vs. AR)	1	30.8	33.2±0.44	11.6±0.41	28.7±0.44

Table 3: Performance on WMT14 De–En and XSum. The two Seq-Drifting rows evaluate results on the dataset ground truth and samples from our support model, respectively. † denotes results taken from Hu et al. (2026).

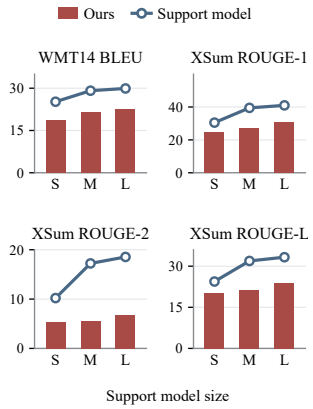

Figure 5: Support model scaling. S, M, and L denote small, medium, and large GPT-2.

Table 4: Performance on SVAMP, ASDiv, and ProofWriter.

Model	SVAMP		ASDiv		ProofWriter	
	ACC↑	Time/sample (ms)↓	ACC↑	Time/sample (ms)↓	ACC↑	Time/sample (ms)↓
AR	27.92±0.36	49.42±4.36	44.97±0.50	31.07±1.18	85.90±0.38	41.90±0.10
NAR	4.72±0.76	2.70±0.04	1.73±0.57	1.30±0.07	0.00±0.00	1.42±0.06
Coconut	17.76±0.01	25.80±0.52	22.59±0.01	26.63±0.13	–	–
Seq-Drifting	13.60±0.50	5.88±0.17	19.11±0.79	1.05±0.03	79.31±0.47	3.68±0.02

although a gap remains relative to the multi-step results. We hypothesize that this gap partly reflects a mismatch between the support-set candidates used during training and the single reference used for evaluation. When evaluated against results sampled from the support model, Seq-Drifting obtains substantially higher scores. Scaling the support model further improves one-step generation quality (Figure 5). This observation motivates future work on using increasingly capable language models to construct support sets, with the broader goal of turning existing LLMs into efficient one-step generators.

Reasoning. We further evaluate Seq-Drifting on math reasoning (SVAMP (Patel et al., 2021) and ASDiv (Miao et al., 2020)) and logical reasoning (ProofWriter (Tafjord et al., 2021)). We scale Seq-Drifting and all the baselines to 0.5B. The implementation and training details are provided in Appendix B.6. Despite using a single forward pass, Seq-Drifting approaches the accuracy of existing latent-reasoning method Coconut (Hao et al., 2025) on the mathematical benchmarks and remains competitive with AR on ProofWriter, while achieving substantially lower per-sample latency. These results show that a drifting field can also be learned from supervision with explicit target answers, extending the one-step framework beyond open-ended generation.

6 Conclusion

In this paper, we presented Seq-Drifting, a framework for directly learning one-step text generation by drifting sequences of continuous token representations. Context-dependent support sets provide multiple plausible targets at each position, while mode-seeking attraction, repulsion, and spherical geometry help preserve coherent and diverse outputs. Across unconditional generation, conditional continuation, translation, summarization, and reasoning, Seq-Drifting provides a competitive one-step operating point and outperforms the evaluated few-step baselines at a fraction of their inference cost. The remaining gap to the strongest multi-step systems motivates future work on stronger support models, richer semantic supervision, and improved token recovery.

AI USE STATEMENT

In this work, we used generative AI tools to edit the manuscript for readability and clarity. All AI-assisted text was reviewed and revised by the authors. The authors take full responsibility for the final content of this work, including all claims, analyses, and conclusions.

ETHICS STATEMENT

This work does not involve human subjects, private or sensitive data, or applications that raise additional ethical concerns. We have reviewed the ICLR Code of Ethics and certify that this submission complies with it.

REPRODUCIBILITY STATEMENT

We describe the model architecture, training procedure, support-set construction, drift objective, datasets, evaluation protocols, and hyperparameters in the main paper and appendix. We provide an anonymized code repository at `https://anonymous.4open.science/r/Seq-Drifting-FEE4/` and include it with the supplementary materials. The repository contains the implementation and instructions needed to reproduce the reported experiments.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pp. 50726–50753, 2025.

Marianne Arriola, Yair Schiff, Hao Phung, Aaron Gokaslan, and Volodymyr Kuleshov. Encoder-decoder diffusion language models for efficient training and inference. *Advances in Neural Information Processing Systems*, 38:15612–15644, 2026.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pp. 2397–2430. PMLR, 2023.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pp. 12–58, 2014.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. Langflow: Continuous diffusion rivals discrete in language modeling. *arXiv preprint arXiv:2604.11748*, 2026.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting, 2026. URL `https://arxiv.org/abs/2602.04770`.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast llms via self-distillation through time. In *International Conference on Learning Representations*, volume 2025, pp. 5007–5045, 2025.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *International Conference on Learning Representations (ICLR)*, 2023.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596, 2023.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025.

Satoshi Hayakawa, Yuhta Takida, Masaaki Imaizumi, Hiromi Wakaki, and Yuki Mitsufuji. Distillation of discrete diffusion through dimensional correlations. In *Forty-second International Conference on Machine Learning*, 2025.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. Elf: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026.

Jaehyeong Jo and Sung Ju Hwang. Continuous diffusion model for language modeling. *Advances in Neural Information Processing Systems*, 38:97394–97430, 2026.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M Boffi, and Jinwoo Kim. One-step language modeling via continuous denoising. *arXiv e-prints*, pp. arXiv–2602, 2026.

Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*, 2025.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35: 4328–4343, 2022.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Rabeeh Karimi Mahabadi, Hamish Ivison, Jaesung Tae, James Henderson, Iz Beltagy, Matthew E Peters, and Arman Cohan. Tess: Text-to-text self-conditioned simplex diffusion. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2347–2361, 2024.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, pp. 975–984, 2020.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 1797–1807, 2018.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. CANDI: Hybrid discrete-continuous diffusion models. In *Forty-third International Conference on Machine Learning*, 2026.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. *Proceedings of machine learning research*, 267: 52584, 2025.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the association for computational linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 1097–1100, 2018.

A Method details and analysis

A.1 Generator architecture

A shared noise vector $\epsilon \in \mathbb{R}^C$ is projected to $T \cdot D$ dimensions, reshaped to (T, D), and combined with learned positional embeddings. Our generator G_θ is a 12-layer pre-LN Transformer with width 768, 12 attention heads, and an FFN width of 3072. The noise dimension is $C = 128$. We use QK-norm on the per-head queries and keys and a LayerScale gate on each residual branch as stabilization mechanisms. For conditional generation, conditioning token embeddings are supplied as a prefix and the output positions access them through ungated cross-attention.

A.2 A conditional fixed-point analysis

We analyze conditions under which matching the generated and data distributions makes the idealized drift field zero. For $q_\theta = p_{\mathrm{data}}$ to be a fixed point of training, the drift V, and hence the loss of Eq. (3), must be zero there. Anti-symmetry $V_{p,q} = -V_{q,p}$ is one way to guarantee this and is how the plain mean-shift field of Deng et al. (2026) does it, but it is a *sufficient* condition, not a necessary one. Seq-Drifting's per-position field is not anti-symmetric, so instead of appealing to that argument we check $V = \mathbf{0}$ at $q_\theta = p_{\mathrm{data}}$ directly.

The field. At position t, with unit embedding $\boldsymbol{e}_t$, the drift of §4 is

$$\boldsymbol{F}_t = \underbrace{\sum_{\boldsymbol{w} \in \mathcal{S}_t} \varphi(\boldsymbol{w}, \boldsymbol{e}_t)(\boldsymbol{w} - \boldsymbol{e}_t)}_{\text{attraction } \boldsymbol{u}_t} - \lambda_{\mathrm{out}} \sum_{\boldsymbol{n} \in N_t^{\mathrm{out}}} \varphi(\boldsymbol{n}, \boldsymbol{e}_t)(\boldsymbol{n} - \boldsymbol{e}_t) - \lambda_{\mathrm{in}} \sum_{\boldsymbol{n} \in N_t^{\mathrm{in}}} \varphi(\boldsymbol{n}, \boldsymbol{e}_t)(\boldsymbol{n} - \boldsymbol{e}_t), \quad (8)$$

with $\varphi(\boldsymbol{a}, \boldsymbol{b}) \propto \exp\big(-d(\boldsymbol{a}, \boldsymbol{b})/(s\,\tau)\big)$ and $d(\boldsymbol{a}, \boldsymbol{b}) = 1 - \boldsymbol{a}^\top \boldsymbol{b}$ the angular kernel, normalized over each set. The attraction pulls $\boldsymbol{e}_t$ toward the support set $\mathcal{S}_t$ of plausible tokens, and the two repulsions push it off the other generations N_t^{out} and the other positions of its own sequence N_t^{in}.

Attraction and rescaled repulsion at matching distributions. At $q_\theta = p_{\text{data}}$ the generator reproduces the data. Suppose each output embedding coincides with a real token embedding, $\boldsymbol{e}_t = \boldsymbol{w}^\star$ with $\boldsymbol{w}^\star \in \mathcal{S}_t$. Importantly, this does not depend on any particular support model. In the mode-seeking limit $\tau \to 0$, a generation already sitting on a supported token has nothing to be pulled toward, so the attraction is zero,

$$\boldsymbol{u}_t = \sum_{\boldsymbol{w} \in \mathcal{S}_t} \varphi(\boldsymbol{w}, \boldsymbol{e}_t)(\boldsymbol{w} - \boldsymbol{e}_t) \;\longrightarrow\; \varphi(\boldsymbol{w}^\star, \boldsymbol{e}_t)(\boldsymbol{w}^\star - \boldsymbol{e}_t) = \boldsymbol{0}, \tag{9}$$

The repulsions then vanish with it, because each is rescaled to the attraction magnitude before entering $\boldsymbol{F}_t$, with coefficient $a_{\text{rms}}/r_{\text{rms}}$ and $a_{\text{rms}} = \text{RMS}(\boldsymbol{u}_t)$, so the repulsion magnitude is $\propto a_{\text{rms}} \to 0$. Under these conditions, $\boldsymbol{F}_t$ is zero at every position and $q_\theta = p_{\text{data}}$ is a fixed point of the idealized field. The attraction-relative rescaling is what lets a strong anti-collapse repulsion coexist with this fixed point, because the repulsion is rescaled by the attraction magnitude and therefore switches off with it.

In practice, the temperature is positive and the decoded token distribution can match the data distribution even when generated representations slightly differ from their corresponding vocabulary embeddings. We now quantify how such discrepancies affect the drift. Let $\boldsymbol{w}_t^\star = \boldsymbol{w}_{\text{tok}_t}$ and consider generated sequences for which $\boldsymbol{w}_t^\star \in \mathcal{S}_t$ is the unique nearest candidate. Since the kernel weights sum to one, the attraction decomposes exactly as

$$\boldsymbol{u}_t = \boldsymbol{w}_t^\star - \boldsymbol{e}_t + \sum_{\boldsymbol{w} \in \mathcal{S}_t \setminus \{\boldsymbol{w}_t^\star\}} \varphi(\boldsymbol{w}, \boldsymbol{e}_t)(\boldsymbol{w} - \boldsymbol{w}_t^\star). \tag{10}$$

The first term is the representation discrepancy, and the second is the finite-temperature correction. For a support set with at least two candidates, define

$$\begin{aligned} \Delta_t &= \min_{\boldsymbol{w} \in \mathcal{S}_t \setminus \{\boldsymbol{w}_t^\star\}} \big[d(\boldsymbol{e}_t, \boldsymbol{w}) - d(\boldsymbol{e}_t, \boldsymbol{w}_t^\star)\big] > 0, \\ b_t &= (|\mathcal{S}_t| - 1) \exp\left(-\frac{\Delta_t}{s_t \tau}\right). \end{aligned} \tag{11}$$

Each other candidate has at most $\exp(-\Delta_t/(s_t\tau))$ times the nearest candidate's kernel weight, giving $1 - \varphi(\boldsymbol{w}_t^\star, \boldsymbol{e}_t) \le b_t/(1 + b_t)$. All vocabulary embeddings have unit norm, so their pairwise distances are at most two. It follows that

$$\big\|\boldsymbol{u}_t - (\boldsymbol{w}_t^\star - \boldsymbol{e}_t)\big\|_2 \le 2\big[1 - \varphi(\boldsymbol{w}_t^\star, \boldsymbol{e}_t)\big] \le \frac{2b_t}{1 + b_t}. \tag{12}$$

For singleton supports, the sole candidate has weight one and we set $b_t = 0$. Define $\|\boldsymbol{a}\|_{\text{RMS}} = (\mathbb{E}\|\boldsymbol{a}_t\|_2^2)^{1/2}$, averaging over generated samples and positions, and write

$$\delta = \big(\mathbb{E}\|\boldsymbol{e}_t - \boldsymbol{w}_t^\star\|_2^2\big)^{1/2}, \qquad \rho_\tau = \left[\mathbb{E}\left(\frac{2b_t}{1 + b_t}\right)^2\right]^{1/2}. \tag{13}$$

The L^2 triangle inequality gives $\|\boldsymbol{u}\|_{\text{RMS}} \le \delta + \rho_\tau$. Under the attraction-relative rescaling above, each rescaled repulsion has RMS no greater than that of the attraction, using the same normalization groups and leaving zero repulsion at zero. Tangent projection cannot increase the force norm, and the geodesic displacement $V_t = goal_t - \boldsymbol{e}_t$ satisfies $\|V_t\|_2 = 2|\sin(\alpha_t/2)| \le \alpha_t \le \|\boldsymbol{F}_t'\|_2$, where α_t is the step angle at position t and zero tangent force gives zero displacement. Applying the triangle inequality to the combined force therefore yields

$$\|V\|_{\text{RMS}} \le \|\boldsymbol{F}\|_{\text{RMS}} \le (1 + \lambda_{\text{out}} + \lambda_{\text{in}})(\delta + \rho_\tau). \tag{14}$$

The loss averaged over samples and positions is bounded by the square of this expression. For a fixed distribution of representations satisfying the support and uniqueness conditions above, $\rho_\tau \to 0$ as $\tau \to 0$ by dominated convergence, since $b_t \to 0$ and $2b_t/(1 + b_t) \le 2$. At positive temperature, the bound remains small when the representation discrepancy and the temperature correction are small enough for the chosen repulsion weights. Thus, when continuous representations do not coincide exactly, matching the decoded distribution can ensure that the drift is bounded.

Algorithm 1: Seq-Drifting training.

Input: generator G_θ, frozen support model p_ψ, normalized token table $\{\boldsymbol{w}_v\}_{v\in\mathcal{V}}$, batch size B, support size k, τ, $\lambda_{\text{out}}, \lambda_{\text{in}}, \alpha_{\max}$.
repeat
  Sample noise $\epsilon_1, \dots, \epsilon_B$ and optional prefixes $\boldsymbol{p}_1, \dots, \boldsymbol{p}_B$.
  Compute one-step outputs $\boldsymbol{h}_{b,1:T} = G_\theta(\epsilon_b, \boldsymbol{p}_b)$ and normalize $\boldsymbol{e}_{b,t} = \boldsymbol{h}_{b,t}/\|\boldsymbol{h}_{b,t}\|$.
  Decode provisional tokens $\hat{x}_{b,t} = \arg\max_v \boldsymbol{e}_{b,t}^\top \boldsymbol{w}_v$.
  Run one causal forward pass of p_ψ on each generated context and form $\mathcal{S}_{b,t}$ from the top-k plausible next tokens.
  for each sample b and position t **do**
    Compute angular attraction $\boldsymbol{u}_{b,t}$ from $\mathcal{S}_{b,t}$.
    Compute cross-sample and intra-sequence repulsions $\boldsymbol{r}_{b,t}^{\text{out}}$ and $\boldsymbol{r}_{b,t}^{\text{in}}$.
    Rescale each repulsion to the attraction magnitude: $\widetilde{\boldsymbol{r}}_{b,t}^{j} = (a_{\text{rms}}/r_{\text{rms}}^{j})\boldsymbol{r}_{b,t}^{j}$ for $j \in \{\text{out}, \text{in}\}$, where the RMS values are computed over the batch and positions; set $\widetilde{\boldsymbol{r}}^{j} = \boldsymbol{0}$ when $r_{\text{rms}}^{j} = 0$.
    Set $\boldsymbol{F}_{b,t} = \boldsymbol{u}_{b,t} + \lambda_{\text{out}}\widetilde{\boldsymbol{r}}_{b,t}^{\text{out}} + \lambda_{\text{in}}\widetilde{\boldsymbol{r}}_{b,t}^{\text{in}}$.
    Project $\boldsymbol{F}_{b,t}$ to the tangent plane and map it to $goal_{b,t}$ by Eq. (7).
  end for
  Update θ with $\nabla_\theta \frac{1}{BT}\sum_{b,t}\|\boldsymbol{e}_{b,t} - \text{sg}(goal_{b,t})\|_2^2$.
until convergence

Figure 6: Seq-Drifting training pseudocode.

A.3 Seq-Drifting algorithm

The pseudocode in Figure 6 summarizes Seq-Drifting training with on-policy support sets constructed by a frozen support model. Each iteration generates token representations, builds context-dependent support sets, and combines attraction and repulsion to form supervision targets on the sphere. The generator is then updated by regressing toward these fixed targets, with optional prefixes enabling conditional generation.

B Additional results

B.1 Self-BLEU

Table 5 reports Self-BLEU (Zhu et al., 2018) for the few-step distilled baselines, at each of their step counts (1, 2, 4), and for Seq-Drifting at a single step. Self-BLEU measures the n-gram overlap among a model's own generations. A fully mode-collapsed generator that emits the same sample every time attains the maximum of 1.0.

B.2 Sampling time across step budgets

Figure 7 reports seconds per sample as a function of the sampling budget for the multi-step baselines. The horizontal dashed line marks the one-step latency of Seq-Drifting. Sampling time grows with the number of denoising steps, while Seq-Drifting remains constant at one step.

B.3 Multi-step baselines

Table 6 gives the full multi-step baselines (FLM, Duo, MDLM, CANDI, RDLM, LangFlow, ELF) at their 1/8/1024 step budgets on both datasets, with Seq-Drifting included for reference. Table 7 gives the multi-step baselines on the *conditional* task at the same step budgets.

Multi-step baseline implementations. For FLM, we use the authors' released checkpoints for LM1B and OpenWebText (Lee et al., 2026). For Duo, MDLM, and CANDI on LM1B, we use the reproduction checkpoints distributed by Lee et al. (2026). On OpenWebText, we use the released Duo and MDLM weights (Sahoo et al., 2024; 2025). For CANDI on OpenWebText, we report the published result from Lee et al. (2026), while no public conditional OpenWebText checkpoint is available. For RDLM, we use the released LM1B checkpoint and its accompanying SDE configuration (Jo & Hwang, 2026). We do not report OpenWebText results because no corresponding

Table 5: **Self-BLEU across the step budget.** Per-dataset Self-BLEU for the few-step distilled models and Seq-Drifting; a fully mode-collapsed generator scores 1.0 (bottom row).

Method	Step	Self-BLEU	
		LM1B	OpenWebText
FMLM	1	0.201	0.249
	2	0.198	0.251
	4	0.204	0.260
Duo+DCD	1	0.042	0.016
	2	0.065	0.064
	4	0.093	0.140
Duo+Di4C	1	0.054	0.082
	2	–	0.105
	4	–	0.190
MDLM+SDTT	1	0.049	0.082
	2	0.061	0.098
	4	0.104	0.177
MDLM+Di4C	1	0.023	0.080
	2	0.039	0.111
	4	0.061	0.204
Seq-Drifting^S	**1**	**0.066**	**0.146**
Seq-Drifting^P	**1**	**0.042**	**0.206**
Mode collapse	–	1.00	1.00

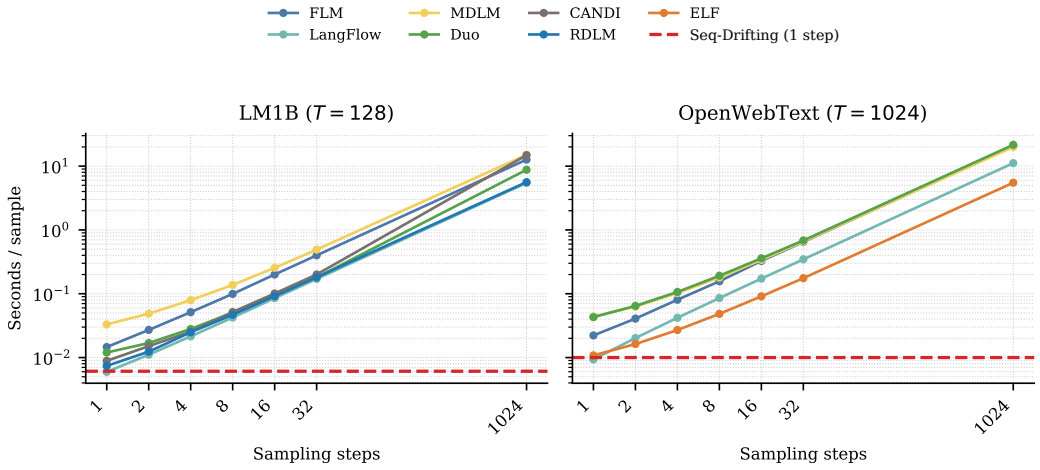

Figure 7: **Sampling time across step budgets.** Seconds per sample on LM1B ($T = 128$) and OpenWebText ($T = 1024$). The x-axis is logarithmic in the number of sampling steps and the y-axis is logarithmic in seconds per sample. The dashed line marks Seq-Drifting.

public checkpoint is available. For LangFlow, we use the authors' released checkpoints (Chen et al., 2026). For ELF, we use the released OpenWebText results and checkpoints (Hu et al., 2026); no corresponding LM1B checkpoint is available.

B.4 Alternative Gen-PPL evaluators

The main comparison reports Gen-PPL under GPT-2 Large. To test whether the results depend on that evaluator, we recompute unconditional Gen-PPL with Pythia-1B and OPT-1.3B in Table 8. All other sampling settings are unchanged.

Table 6: **Unconditional generation: multi-step models** across the step budget, with Seq-Drifting at a single pass. LM1B (T=128) and OpenWebText (T=1024). A dash indicates that no corresponding released checkpoint or published reference result was available for that dataset–model setting.

Method	Step	LM1B (T=128)		OpenWebText (T=1024)	
		Gen-PPL↓	Entropy	Gen-PPL↓	Entropy
FLM	1	6.27	1.15	27.54	1.53
	8	215.84	4.24	453.49	5.20
	1024	85.24	4.29	63.97	5.35
Duo	1	1027.77	4.34	4007.98	5.80
	8	121.37	4.27	196.45	5.57
	1024	85.97	4.30	78.11	5.54
MDLM	1	1348.21	4.37	4180.42	5.82
	8	177.21	4.33	785.33	5.87
	1024	69.39	4.30	42.36	5.30
CANDI	1024	108.07	4.35	143.13	5.71
RDLM	1	22.70	3.20	–	–
	8	229.32	4.01	–	–
	1024	274.94	4.35	–	–
LangFlow	1	1.69	0.30	14.53	1.00
	8	536.60	4.38	514.24	5.80
	1024	121.73	4.31	36.53	5.25
ELF	1	–	–	2.97	1.99
	8	–	–	70.95	5.25
	1024	–	–	7.73	4.74
Seq-Drifting^S	**1**	**93.81**	**4.21**	**183.98**	**4.95**
Seq-Drifting^P	**1**	**36.01**	**4.42**	**128.33**	**5.08**

B.5 Translation and summarization setup

For WMT14, the input contains a 64-token source sequence and the model generates a 64-token target sequence. For XSum, the input contains up to 1024 source tokens and the model generates a 64-token summary. We report corpus BLEU (Papineni et al., 2002) for WMT14 and ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) for XSum.

For Seq-Drifting, we first train an autoregressive model from scratch on each task's training split. In the small support-model setting, this model is a 12-layer pre-LN Transformer with hidden width 768, 12 attention heads, FFN width 3072, and the same tokenizer and context lengths used by the corresponding experiment. It uses a causal attention mask and is trained with the standard next-token cross-entropy objective. We then train Seq-Drifting using the frozen autoregressive model's token distributions as its on-policy support signal, without initializing from a pretrained language model. At evaluation time, we compare generated outputs against both the dataset ground-truth references and samples generated by the autoregressive model, using the same task metrics in both cases.

B.6 Reasoning implementation details

For mathematical reasoning, we train on GSM8K-Aug (Deng et al., 2023) and evaluate out-of-domain generalization on SVAMP and ASDiv. Ground-truth solution sequences define the support set for the attraction objective. For logical reasoning, we use the binary (true/false) subset of ProofWriter, training on the combined D0–D5 training split and evaluating on the corresponding test split; unknown-labeled examples are excluded. Seq-Drifting uses a 0.5B-parameter backbone with the Qwen2.5 architecture. We compare it with autoregressive (AR), non-autoregressive (NAR), and latent-reasoning baseline Coconut (Hao et al., 2025). We report answer accuracy and wall-clock time per sample for each benchmark in Table 4.

- **Coconut** performs latent reasoning by feeding continuous thought representations back into the model's hidden states instead of emitting explicit chain-of-thought tokens.

Table 7: **Conditional generation: multi-step models** across the step budget, with Seq-Drifting at a single pass. A 64-token prefix, then a continuation (64→64 on LM1B, 64→512 on OWT). A dash indicates that no corresponding released checkpoint or published reference result was available for that dataset–model setting.

Method	Step	64→64		64→512	
		Gen-PPL↓	Entropy	Gen-PPL↓	Entropy
FLM	1	1.95	0.83	201.21	2.18
	8	6.00	1.81	522.85	3.63
	1024	9.47	2.19	304.01	3.44
Duo	1	1103.17	3.83	7616.55	3.86
	8	99.98	3.78	181.91	3.81
	1024	72.04	3.81	84.65	3.82
MDLM	1	1064.26	3.83	6026.52	3.85
	8	147.58	3.82	490.74	3.86
	1024	58.64	3.80	61.71	3.79
CANDI	1024	93.09	3.84	–	–
RDLM	1024	72.79	3.21	–	–
LangFlow	1	2.94	0.32	72.60	0.74
	8	242.77	3.87	743.79	3.78
	1024	64.05	3.81	83.30	3.31
ELF	1	–	–	4.50	1.82
	8	–	–	91.49	4.99
	1024	–	–	10.59	4.49
Seq-Drifting^S	**1**	**79.13**	**3.74**	**119.13**	**4.10**
Seq-Drifting^P	**1**	**76.16**	**4.40**	**168.34**	**4.78**

Table 8: **Unconditional Gen-PPL under alternative evaluators.** Gen-PPL computed with Pythia-1B and OPT-1.3B for LM1B (T=128) and OpenWebText (T=1024).

Method	Step	LM1B		OpenWebText	
		Pythia-1B	OPT-1.3B	Pythia-1B	OPT-1.3B
FMLM	1	94.75	97.29	170.10	152.67
	2	87.55	88.58	134.89	122.45
	4	79.86	80.37	109.44	100.41
Duo+DCD	1	829.59	849.31	4654.30	4401.95
	2	435.40	453.47	905.38	792.78
	4	254.97	260.45	246.35	219.22
Duo+Di4C	1	–	–	1029.73	1061.16
	2	–	–	736.75	728.92
	4	–	–	252.60	251.94
MDLM+SDTT	1	919.99	935.31	1024.45	1086.82
	2	397.09	414.49	802.18	809.25
	4	161.86	162.99	306.35	307.63
MDLM+Di4C	1	1088.63	1120.40	1025.45	1067.07
	2	601.01	626.70	702.27	695.89
	4	266.63	273.17	216.24	215.50
Seq-Drifting^S	**1**	**97.04**	**98.70**	**130.48**	**151.38**
Seq-Drifting^P	**1**	**36.07**	**43.07**	**110.36**	**119.68**

B.7 TRAINING DYNAMICS

Figure 8 tracks Gen-PPL and unigram entropy over training for the conditional runs. The first 200k steps are the unconditional warm-up phase, during which the context is filled with a constant token

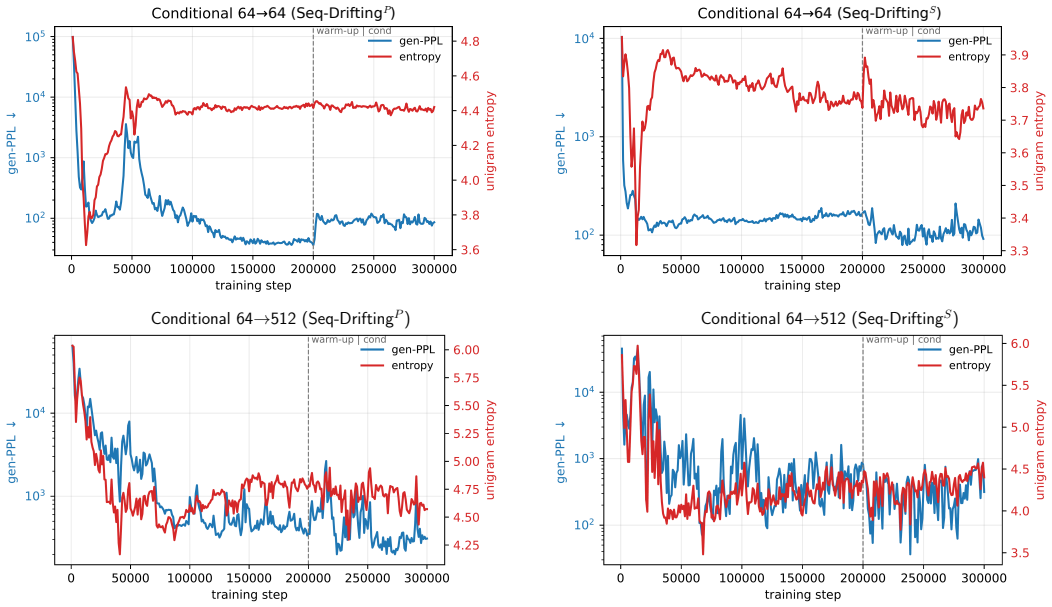

Figure 8: **Training dynamics of the conditional runs.** Gen-PPL (log scale, left axis) and unigram entropy (right axis) versus training step, for Seq-Drifting^P 64→64 (top left), Seq-Drifting^S 64→64 (top right), Seq-Drifting^P 64→512 (bottom left), and Seq-Drifting^S 64→512 (bottom right). All panels show training through 300k steps for both output lengths. The dashed line marks the warm-up→conditional switch at 200k steps.

under normal attention. The drift reaches coherent English and drives Gen-PPL down by orders of magnitude; real prefix conditioning then turns on and perplexity keeps falling.

B.8 Off-policy positives

The *off-policy* ablation of §5.4 constructs support sets from corpus retrieval instead of language-model predictions. We precompute a memory bank of the token n-gram embeddings of a set of real sequences, and at each generated position we take as the support set $\mathcal{S}_t$ the tokens that real sequences place at the position where they best match the current generation, ranked by the position-aligned min-n-gram distance of §4.2. Everything downstream, the angular attraction, the two repulsions, and the geodesic update, is unchanged. The one difference from the on-policy signal is that these targets are *fixed* by the data and do not follow the generator.

Observed training behavior. Figure 9 tracks this run at length 128. During the temperature warm-up, over which the attraction temperature τ anneals from 1.0 to 0.1 across the first 10k steps, Gen-PPL falls as the soft, mode-covering attraction pulls the batch toward real n-grams. Once τ reaches its sharp steady value at 10k, each generation commits hard to its single nearest real n-gram. Because those targets are fixed and do not adapt, the sharp commitment lets the anti-collapse repulsion over-scatter the batch faster than the fixed attraction can pull it back, which sets up a limit cycle. The repulsion over-scatters the batch, so entropy and Gen-PPL spike. The samples are then far apart, the repulsion weakens, and the attraction pulls them back toward real tokens, so entropy and Gen-PPL fall. Once re-consolidated the repulsion strengthens again and the cycle repeats, which is the sawtooth in Figure 9. The cycle does not close, each swing ends higher than the last, so Gen-PPL and entropy trend upward overall and the run moves toward higher-entropy text with reduced global coherence. The on-policy support model avoids this this because it scores the generation's own prefix, so its targets follow the generator and the attraction stays balanced against the repulsion.

B.9 Qualitative samples

The following examples are produced in a single forward pass and truncated for display.

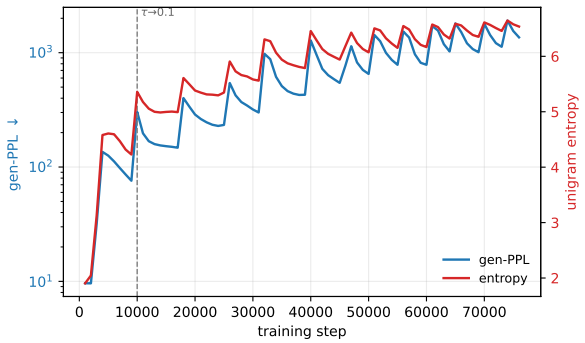

Figure 9: **Off-policy training dynamics** (unconditional, length 128). Gen-PPL (log scale, left axis) and unigram entropy (right axis) versus training step for the off-policy positives ablation. The dashed line marks the end of the temperature warm-up at 10k steps, where τ reaches 0.1 and the run begins to oscillate and degrade.

Seq-Drifting^{P}

```
entropy:  4.30 Gen-PPL: 36.01

It will provide an excellent overview of a wide variety of topics:
"Innovations & Challenges of the Future," "A History of Technology"
and, the latest developments and advancements that the technology has
in general and the development and future capabilities in specific
applications.  This is one of our largest conferences ever organized
for industry-leading attendees in North American technology industry,
which will give you more insight as you go to work with technology.
About the Author:
Klaus K. Krieg was awarded his doctor, ...
```

Seq-Drifting^{S}

```
entropy:  4.21 Gen-PPL: 93.81

A spokeswoman at the Brown's Downing Street offices said:  "There it
would not happen here at home, will be able to make up his mind
whether that's right another very me.  i guess he has been very, but
if he did have been opportunity, if he has been very very me.  the so
far as well ...  and this time around there, if he did.  i think
terms people who knows about what happened on afternoon, ...
```

Seq-Drifting WMT

Source: Wir sind stolz auf unsere Leistung, aber wir wollen jedes Spiel gewinnen.
Reference: We are proud of our performance, but we want to win every game.
Prediction: We are proud of our performance, but we want to win every game.

Seq-Drifting XSum

Source: Police said officers from the North West counter terrorism unit searched an address on Peakdale Avenue, Crumpsall, Manchester on Friday. The suspect, 26, was arrested the following day on suspicion of offences under the Terrorism Act. Police would not comment on the nature of the alleged offence but said it believed it caused "no threat" to the community.
Reference: A man is being held in Manchester on suspicion of terrorism offences.
Prediction: A man has been arrested on suspicion of terrorism offences

Seq-Drifting Math Reasoning

Question: A group of students are preparing for a water balloon fight. Each student needs 5 balloons. If there are 12 students, how many balloons are needed in total?
Prediction: <<5*12=60>>
60

Seq-Drifting ProofWriter

Question: Theory: Bob is kind. Bob is nice. Bob is white. Charlie is kind. Charlie is white. Erin is red. Erin is rough. Harry is blue. Harry is kind. Harry is red. Blue people are nice. All white people are red. If someone is white and blue then they are nice. All rough people are red. If someone is smart then they are blue. If someone is kind then they are red. If someone is nice then they are rough. If someone is red then they are smart. If someone is red and rough then they are nice.
Query: Bob is not nice. Give the proof in order and finish with #### true or #### false.
Prediction: Fact: Bob is nice.
false