DEPENDENCY-AWARE MASKING FOR PARALLEL DECODING IN MASKED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked Diffusion Language Models (MDLMs) offer a parallel alternative to autoregressive models by enabling parallel text generation. However, simultaneously unmasking multiple tokens relies on a strict conditional independence assumption. Standard training and sampling are both structure-agnostic, so strongly coupled syntactic pairs can be decoded jointly, introducing significant factorization errors that degrade text quality. To address this at training time, we propose Dependency-Aware Masking, a non-intrusive training formulation that utilizes linguistic dependency trees to replace the standard uniform noise schedule with a depth-aware soft prior. This biases the model to safely reconstruct conditionally independent peripheral modifiers before resolving the densely coupled syntactic core. At inference time, we pair it with a multi-token sampler that unmasks independent sets of an attention-derived interaction graph, avoiding the joint decoding of strongly coupled tokens. Extensive evaluations across MDLM and BD3-LM architectures demonstrate that our framework consistently outperforms uniform-masking baselines on unconditional generation, establishing a superior trade-off between generation quality and decoding efficiency.

1 INTRODUCTION

Autoregressive (AR) models (Brown et al., 2020; Touvron et al., 2023) have long dominated natural language generation by modeling text as a sequence of tokens generated strictly left-to-right. While highly effective, this causal formulation fundamentally limits generation speed, as decoding must proceed one token at a time. As the demand for inference efficiency grows, enabling parallel decoding has become a crucial goal. Discrete diffusion language models (Austin et al., 2021; Lou et al., 2024), and in particular their masked (absorbing-state) variant (MDLMs) (Sahoo et al., 2024; Ou et al., 2025; Nie et al., 2024), sidestep this bottleneck. They start from a fully masked sequence and unmask multiple tokens per step, decoupling the number of forward passes from sequence length.

However, parallel decoding in MDLMs introduces a key challenge. To decode multiple tokens simultaneously, the model must rely on a *conditional independence assumption* (Gu et al., 2017; Liu et al., 2024; Xu et al., 2024; Wu et al., 2026; Kim et al., 2026b). Specifically, the tokens being unmasked in the same step are independent of one another given the previously generated context. Standard multi-token samplers are structure-agnostic, selecting tokens to unmask uniformly at random or by per-position confidence or entropy (Sahoo et al., 2024; Chang et al., 2022; Kim et al., 2025). When such samplers group strongly coupled tokens into the same step (e.g., jointly predicting "San" and "Francisco"), the independence assumption fails catastrophically. This causes the factorized output to diverge from the true joint distribution, producing locally incoherent generations.

To resolve this, we must ask: *what is the optimal trajectory for parallel generation?* Natural language is a deeply hierarchical system mapped by linguistic dependency trees (De Marneffe & Manning, 2008; Nivre et al., 2016). In these trees, words are organized by structural depth into distinct syntactic layers. Crucially, tokens that reside in the same peripheral layer or on separate branches (e.g., distinct adjectives modifying separate nouns) are only weakly coupled. Therefore, rather than using arbitrary schedules, a good parallel decoding strategy should group tokens by their structural layers. By parallelizing the generation of conditionally independent modifiers and structurally separating them from the densely coupled core, the model naturally avoids dependency collisions and minimizes factorization errors. A structured sampler, however, only changes how the model is decoded, not

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.

what it has learned. Aligning the training corruption process with the same structural prior improves the model itself, yielding gains under every sampler we evaluate.

Motivated by this insight, we introduce a **Dependency-Aware Framework** that systematically aligns the MDLM generation trajectory with the linguistic dependency tree. Our approach is completely non-intrusive: the model architecture and underlying denoising objective remain unchanged. We intervene solely at the training masking schedule and the inference sampling algorithm:

First, we propose **Dependency-Aware Masking** to inject an explicit syntactic bias into the forward corruption process. Using structural depths extracted from an off-the-shelf dependency parser (Qi et al., 2020; Honnibal et al., 2020), we replace the standard uniform masking schedule with a token-specific soft prior. By assigning higher masking probabilities to the shallow structural core and lower probabilities to deep modifiers, we probabilistically bias the model to learn a bottom-up trajectory that tends to reconstruct peripheral context first and recover the central core only when sufficient surrounding context is established.

Second, to exploit this prior at inference time, we build on DAPD (Kim et al., 2026a) and design a **Dependency-Aware Multi-Token Sampler** tailored to it. The sampler constructs an interaction graph from the model's internal attention weights and unmasks an *independent set* of this graph in parallel, avoiding structural collisions without prioritizing the core tokens that our prior defers.

We evaluate our method on two backbones, MDLM (Sahoo et al., 2024) and BD3-LM (Arriola et al., 2025), using the LM1B and OpenWebText datasets. Empirical results demonstrate that our dependency-aware framework consistently improves upon uniform-masking baselines on unconditional text generation, achieving a superior trade-off between generation quality and sampling efficiency.

2 Related Work

Masked Discrete Diffusion Language Models. Diffusion-based text generation spans continuous and discrete spaces. Early work lifts text into embeddings or latent spaces (Li et al., 2022; Gong et al., 2023; Strudel et al., 2022; Dieleman et al., 2022; Gulrajani & Hashimoto, 2023). Discrete approaches build directly on tokens, starting from D3PM (Austin et al., 2021) and a range of follow-ups (Hoogeboom et al., 2021; He et al., 2023; Campbell et al., 2022; Sun et al., 2022; Chen et al., 2022; Wu et al., 2023; Zheng et al., 2023; Gat et al., 2024), among which masked diffusion language models have attracted particular attention for their strong empirical performance (Shi et al., 2024; Lou et al., 2024; Ou et al., 2025; Sahoo et al., 2024; Nie et al., 2024). In particular, MDLM (Sahoo et al., 2024) formulates a simplified Rao-Blackwellized ELBO that ties masked-token prediction to a principled generative process, while BD3-LM (Arriola et al., 2025) partitions sequences into contiguous blocks and runs diffusion within each. Later extensions interpolate mask and uniform noise (von Rütte et al., 2025), replace binary masks with soft embedding blends (Hersche et al., 2026), and enable variable-length generation via insertion (Kim et al., 2026b). More recent scaling efforts (Gong et al., 2025; Nie et al., 2025; Ye et al., 2025; Zhu et al., 2025a;b) reach billion-parameter scale, with extensions to code (Gong et al., 2026), multimodal (Yang et al., 2025) and commercial deployment (Khanna et al., 2025; Song et al., 2025; DeepMind, 2025; Bie et al., 2025). Our method is situated directly in this line: rather than changing the backbone or training objective, we modify only the per-token masking probability during training and decode with a dependency-aware sampler.

Token-Aware and Non-Uniform Masking in Diffusion Language Models. A growing line of work derives a per-token masking signal from input properties rather than uniform corruption. Distributional or information-theoretic signals include frequency (He et al., 2023; Kosmopoulou et al., 2025), entropy- or information-aware schedules (Wang et al., 2023; Koh et al., 2025), LLM-annotated information density (Ma et al., 2026), TF-IDF (Chen et al., 2023), and attention-derived importance (Dat et al., 2025). Beyond distributional signals, TRIMS (Chen et al., 2026) uses autoregressive-teacher difficulty, TreeDiff (Zeng et al., 2025) uses abstract syntax trees for code, and GenMD4 (Shi et al., 2024) learns per-token-value mask rates. We instead use natural-language dependency-tree depth, an explicit syntactic signal computed offline from a parser, complementing existing approaches.

Samplers for Masked Diffusion Language Models. Beyond the random ancestral sampler in MDLM (Sahoo et al., 2024), semi-autoregressive decoding (Arriola et al., 2025; Nie et al., 2025) processes blocks left to right, and uncertainty-driven strategies prioritize positions by confidence,

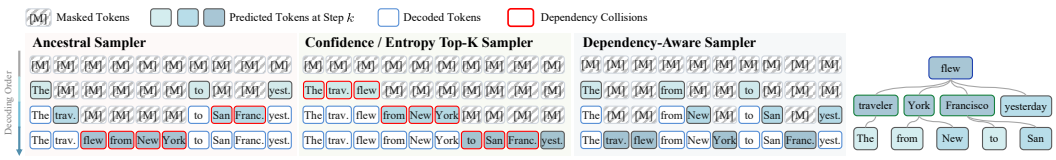


Figure 1: **Comparison of multi-token generation strategies during parallel decoding. Left & Middle:** Structure-agnostic samplers, such as Ancestral and Confidence/Entropy-Top-K Sampler, frequently select structurally dependent tokens in the same generation step (highlighted in red). This violates the conditional independence assumption of parallel decoding, leading to factorization errors. **Right (idealized):** A structure-driven schedule that follows the true dependency tree unmasks an independent set at each step, proceeding bottom-up from peripheral leaves to the syntactic core, so that strongly coupled tokens are never decoded together. In practice, our sampler approximates this schedule with an interaction graph estimated from the model's attention (Sec. 4.3).

entropy, or margin (Chang et al., 2022; Nie et al., 2025; Ye et al., 2025; Kim et al., 2025). EB-Sampler (Ben-Hamu et al., 2025) bounds cumulative entropy for multi-token decoding, and Fast-dLLM (Wu et al., 2026) uses a confidence-threshold rule that we adopt for our second phase. A separate line builds dependency structures over masked positions to avoid co-decoding coupled tokens, either inferred from the backbone's self-attention (Kim et al., 2026a; Luo et al., 2026; Zhou et al., 2026; Azangulov et al., 2026) or from a lightweight learned predictor (Ringel et al., 2026). These methods leave the training of the diffusion model itself unchanged; our contribution is instead a training-time prior that is complementary to them. We pair it with a sampler built on DAPD (Kim et al., 2026a) (Sec. 4.3).

3 Background

Continuous diffusion models (Ho et al., 2020; Song et al., 2021; Kingma et al., 2021; Liu et al., 2023; Lipman et al., 2022) have achieved significant success by gradually corrupting data with noise and learning a reverse denoising process. Early attempts to extend this paradigm to discrete text sequences modeled the forward process using dense categorical transition matrices (Austin et al., 2021; Hoogeboom et al., 2021). However, learning the reverse process under this formulation requires computationally expensive matrix multiplications over the full vocabulary size K.

Masked Diffusion Language Models (MDLM) (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025) significantly simplify discrete diffusion by abandoning dense transition matrices in favor of a linear interpolation between the clean data $\mathbf{x}$ and an absorbing `[MASK]` token $\mathbf{m}$. Assuming the forward process factorizes independently across tokens, the marginal distribution at diffusion time t is defined as:

$$q(\mathbf{z}_t \mid \mathbf{x}) = \text{Cat}(\mathbf{z}_t; \alpha_t \mathbf{x} + (1 - \alpha_t)\mathbf{m}) \tag{1}$$

where $\alpha_t \in [0, 1]$ is a monotonically decreasing noise schedule shared across all token positions. Under this absorbing formulation, the posterior computation for any intermediate step $s < t$ collapses into a highly efficient piecewise form:

$$q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) = \text{Cat}\left(\mathbf{z}_s; \frac{(1 - \alpha_s)}{1 - \alpha_t}\mathbf{m} + \frac{(\alpha_s - \alpha_t)}{1 - \alpha_t}\mathbf{x}\right)$$

During generation, the clean data $\mathbf{x}$ is unobserved. Therefore, the neural network is parameterized to directly predict $\mathbf{x}$ from the masked latent $\mathbf{z}_t$. This "$\mathbf{x}$-prediction" objective bypasses the complex transition evaluations entirely.

Iterative Unmasking and Generation. At inference time, generation begins with a sequence of pure `[MASK]` tokens. The sequence is decoded iteratively: at each step, the model predicts the underlying clean tokens, and a subset of these tokens is permanently "unmasked" (fixed) based on a predefined schedule. In standard MDLMs, the selection of which tokens to unmask in parallel is typically structure-agnostic, relying on purely random selection or model confidence/entropy heuristics.

This framework can also be combined with autoregressive modeling, such as in Block Discrete Diffusion (Arriola et al., 2025), where tokens within individual chunks are decoded in parallel via MDLM, but the transitions across chunks follow a causal structure.

4 METHOD

4.1 STRUCTURAL INDEPENDENCE IN PARALLEL DECODING

Parallel Generation. Mask-based language models can predict arbitrary subsets of tokens conditioned on others. This process can be described by a *block schedule* $\mathcal{B} = (B_1, \ldots, B_T)$, which partitions the token indices $1, \ldots, N$ into T disjoint blocks. At each step t, all tokens in block B_t are generated simultaneously, conditioned on all previously generated tokens $\mathbf{x}_{<B_t}$. Because the tokens within B_t are generated at the exact same time, they cannot attend to one another. Consequently, this parallelization relies on a conditional independence assumption within each block: the model must assume that tokens within B_t are independent given the context, factorizing the distribution into the product of individual token probabilities:

$$\hat{P}_{\mathcal{B}}(\mathbf{X}) := \prod_{t=1}^{T} \prod_{j \in B_t} p_\theta(\mathbf{x}_j \mid \mathbf{x}_{<B_t}). \tag{2}$$

The Problem of Generation Order. While computationally necessary for parallelization, this conditional independence assumption is fundamentally flawed for natural language, which is rich in local dependencies. Approximating the true joint distribution with Eq. (2) inherently introduces a factorization error. Crucially, the magnitude of this error is determined completely by the chosen schedule $\mathcal{B}$. If strongly coupled tokens (e.g., predicting "San" and "Francisco" simultaneously) are assigned to the same block B_t, the independence assumption fails catastrophically. Conversely, if the schedule is designed so that the tokens within any given block are largely independent given $\mathbf{x}_{<B_t}$, this error is minimized. Therefore, determining the optimal generation trajectory is not merely a mechanism for speed, but a critical requirement for preserving sample quality.

Structural Independence via Tree Depth. To design a schedule that minimizes this factorization error, we can leverage the inherent hierarchy of natural language. Natural language is not merely a flat, left-to-right sequence; it is a deeply hierarchical system. This hierarchy can be explicitly mapped using linguistic dependency trees, which capture the structural relationships between words in a sentence. For instance, two distinct adjectives modifying separate nouns in a sentence do not need to communicate to be accurately predicted. As illustrated in Figure 1 (right), leaf nodes such as "The" and "yesterday" share no direct edge and are safer to decode together. Conversely, structure-agnostic strategies like ancestral or confidence-based samplers (Figure 1, left and middle) frequently unmask strongly coupled pairs (e.g., "San" and "Francisco") in the same block, resulting in catastrophic dependency collisions. Therefore, if we bias the parallel decoding schedule so that tokens of similar dependency depth tend to be grouped into the same block B_t, we naturally isolate strongly coupled syntactic pairs into different generation steps. As illustrated in Figure 1, this structure-driven schedule provides a practical proxy for encouraging conditional independence within each decoding block in Eq. (2), thereby reducing factorization error without sacrificing decoding speed.

To practically achieve this structure-driven generation trajectory, we inject dependency information into the MDLM architecture at both training and inference **(Figure 2)**. First, we modify the forward corruption process to apply a depth-aware masking schedule, explicitly biasing the model to learn a structurally sound generation order (Sec. 4.2). Second, building on DAPD (Kim et al., 2026a), we design a dependency-aware multi-token sampler tailored to this prior, which avoids unmasking strongly coupled tokens simultaneously (Sec. 4.3). A key practical advantage of this approach is its minimal implementation overhead. The underlying model architecture and denoising objective remain unchanged, and the required dependency trees can be cheaply extracted offline using off-the-shelf parsers.

4.2 DEPENDENCY-AWARE MASKING

To inject syntactic bias into the MDLM framework, we modify the forward corruption process during training. As established in Eq. (1), standard MDLMs employ a position-uniform noise schedule α_t, yielding a uniform masking probability of $1 - \alpha_t$ across all positions. This structure-agnostic policy ignores the distinct syntactic roles of different tokens.

A natural intuition to address this is to completely replace the uniform schedule with a strictly deterministic, structure-driven schedule. Under such a regime, the forward process would rigidly follow the dependency tree: shallow tokens (the syntactic core) would always be masked first, and

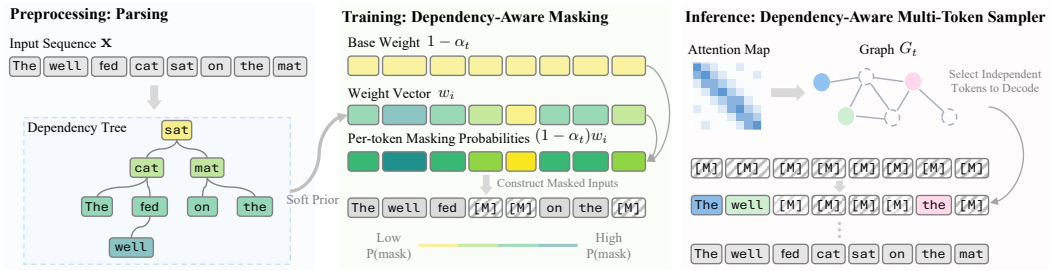


Figure 2: **Overview of the Dependency-Aware Framework. Left (Preprocessing):** Off-the-shelf parsers extract a linguistic dependency tree from the input text, categorizing tokens by structural depth. **Center (Training):** We introduce a depth-aware soft prior by deriving a weight vector w_i from the tree. This vector scales the base masking probability $(1-\alpha_t)$, assigning higher masking rates to the densely coupled syntactic core and lower rates to peripheral modifiers. This biases the model to learn a structurally safe, periphery-to-core generation trajectory. **Right (Inference):** To safely exploit this prior during parallel decoding, our Dependency-Aware Sampler dynamically builds an interaction graph G_t using the model's internal attention map. By extracting an independent set from this graph, we avoid unmasking strongly coupled tokens in the same step.

deep tokens (peripheral modifiers) would be masked last. This forces the model to learn a "bottom-up" generation trajectory strictly, predicting the shallow core solely conditioned on the deep modifiers.

However, a strictly deterministic schedule is likely to degrade performance in practice, as restricting the diversity of masked states during training can prevent the model from learning the robust, arbitrary-order representations that make standard MDLMs effective.

Rather than rigidly enforcing a tree-based order, a more effective approach is to inject syntax as a *soft prior*. We aim to merge the structural trajectory of the dependency tree with the arbitrary-order robustness of standard MDLM scheduling. We achieve this by replacing the uniform schedule with a position-specific masking probability:

$$1-\alpha_t^i = (1-\alpha_t)w_i$$

where w_i is a depth-aware weight. Following the MDLM formulation in Eq. (1), the resulting per-token forward marginal distribution becomes:

$$q(\mathbf{z}_t^i \mid \mathbf{x}^i) = \mathrm{Cat}\big(\mathbf{z}_t^i; [1-(1-\alpha_t)w_i]\mathbf{x}^i + (1-\alpha_t)w_i\mathbf{m}\big)$$

We compute the token-specific modulation weights w_i that inversely scale with token depth, ensuring that shallower tokens (the syntactic core) receive higher baseline weights than deeper tokens (peripheral modifiers). The strength of this depth signal in w_i is controlled by an interpolation coefficient $\lambda \in [0,1]$, where $\lambda = 0$ recovers standard MDLM. The full parameterization of w_i and λ is detailed in Appendix A.1.

Dependency Extraction. We extract this structural signal offline using highly optimized, off-the-shelf dependency parsers (e.g., spaCy (Honnibal et al., 2020) and Stanza (Qi et al., 2020)), broadcasting the resulting word-level depths directly to their constituent subword tokens. Because this extraction is a one-time preprocessing step relying on ready-to-use tooling, it introduces negligible computational overhead, effectively equipping the training pipeline with a rich syntactic prior for free.

4.3 Dependency-Aware Multi-Token Sampler

The masking strategy detailed in Sec. 4.2 encodes a soft syntactic prior into the model's parameters. While preserving the capacity to decode in arbitrary orders, the model is probabilistically biased to reconstruct peripheral modifiers early and resolve the syntactic core conditioned on that context.

To exploit this learned trajectory at inference time, standard structure-agnostic heuristics (such as ancestral sampling) are insufficient. Furthermore, naively extending a standard confidence-based unmasking strategy (Chang et al., 2022; Wu et al., 2026) to multiple tokens risks simultaneously decoding strongly coupled positions, which violates the conditional independence assumption and degrades generation quality. Following the core mechanism of DAPD (Kim et al., 2026a), we unmask an independent set of an attention-derived interaction graph, and design the selection rule around the periphery-to-core order learned by our prior: (i) candidates are ranked by confidence alone, since degree-weighted priority would favor the core tokens the prior defers; and (ii) commits per step are

capped, as the independent set can occasionally grow very large. Detailed implementation parameters for both phases are provided in Appendix A.2.

Phase 1: Attention-Guided Independent Unmasking. In the early stages of decoding, the sequence is densely masked and the risk of generating strongly coupled tokens simultaneously is high. At each step t, let M_t denote the set of currently masked positions. We construct a dynamic interaction graph $G_t = (M_t, E_t)$ over these positions to map structural dependencies. To determine the edge set E_t, we extract the attention weights a_{ij} from the model's top layers and define a symmetrized mutual attention score $s_{ij} = \frac{1}{2}(a_{ij} + a_{ji})$. An edge $(i, j) \in E_t$ is drawn if s_{ij} exceeds a threshold δ_t.

To ensure safe parallel generation, we select a parallel subset $S_t \subseteq M_t$ that forms an *independent set* within G_t, meaning no two selected tokens share a connecting edge ($\forall i, j \in S_t, (i, j) \notin E_t$). The attention graph simply acts as a topological safeguard to prevent strongly coupled modifiers from colliding in the same forward pass (analyzed in Appendix E).

Phase 2: Graph-Free Confidence Unmasking. As decoding progresses and Phase 1 iteratively resolves the most confident, independent tokens, the remaining masked sequence naturally exhibits higher structural interdependence. Let $r_t = |M_t|/N$ denote the current mask ratio. Once r_t drops below a critical transition threshold $r^\star$, large-scale parallelization is no longer suitable, as the remaining structural core must be carefully resolved based on the established context.

By transitioning from an attention-aware graph to a confidence-threshold rule that only unmasks highly confident tokens, the sampler preserves generation quality in the final steps.

5 Experiments

We evaluate our dependency-aware framework to answer three key questions: (1) Does our method improve unconditional generation quality across different diffusion backbones and inference budgets (Sec. 5.1)? (2) Does the model empirically learn the intended periphery-to-core generation order (Sec. 5.2)? (3) How sensitive is the framework to alternative masking heuristics and hyperparameter choices (Sec. 5.3 and Sec. 5.4)?

5.1 Main Results on Masked Diffusion Language Models

Experimental setup. We evaluate our method on two masked diffusion language modeling backbones, MDLM (Sahoo et al., 2024) and BD3-LM (Arriola et al., 2025), using the One Billion Word Benchmark (LM1B) (Chelba et al., 2013) and OpenWebText (OWT) (Gokaslan & Cohen, 2019) datasets. Both backbones adopt the standard DiT architecture with 110M parameters; we follow the released training configurations of MDLM and BD3-LM exactly, training for 1M steps on four H100 GPUs with global batch size 512. Dependency trees are extracted offline using spaCy (Honnibal et al., 2020) (with Stanza (Qi et al., 2020) as a sensitivity check in Sec. 5.4). Full dataset, training, and sampler details are provided in Appendix B.

To evaluate unconditional generation quality, we report *generative perplexity* (Gen. PPL, lower is better), computed by scoring unconditional samples under a held-out GPT-2-Large evaluator (Radford et al., 2019), following standard protocol in prior work (Sahoo et al., 2024; Arriola et al., 2025). Evaluation details are in Appendix C.

Results on unconditional text generation. Table 1 reports unconditional generation results on LM1B and OpenWebText. To isolate the training-side effects, we apply our dependency-aware sampler to both our method and the baselines. Across both datasets and backbones (MDLM and BD3-LM), dependency-aware masking consistently improves generative perplexity over uniform baselines. This isolates the contribution of the training-side intervention, showing that the gain is not solely an artifact of the dependency-aware sampler.

Performance across NFE budgets. We next evaluate performance across varying inference budgets by sweeping the sampler transition threshold $r^\star$ to vary NFE (Sec. 4.3). Figure 3 shows our method consistently outperforms the uniform baseline, demonstrating that training-side structural bias yields a superior quality-efficiency trade-off independent of compute budget (wall-clock times in Appendix F.3).

Robustness across samplers. Does Dependency-Aware Masking require our specific sampler? Table 2 evaluates our models across three samplers. Our Shallow-First approach achieves the

Table 1: **Dependency-aware masking improves unconditional generation across both backbones and both datasets.** The lower block is the comparison of interest: our dependency-aware multi-token sampler is held *fixed* for every row, so the only difference within each backbone pair is the training-time masking policy. The upper block lists each backbone with its own native ancestral sampler as a reference point; for BD3-LM ($L' = 16$) this means that several tokens of each block are committed per step. All decoding configurations operate at matched NFE budgets ($\sim$80 for LM1B and $\sim$512 for OpenWebText). Lower is better for Gen. PPL; entropy is reported as a diversity reference.

		LM1B		OpenWebText	
Backbone	**Training-time masking**	**Gen. PPL↓**	**Entropy**	**Gen. PPL↓**	**Entropy**
Reference: each backbone with its native ancestral sampler					
MDLM	Uniform	74.56	4.25	37.23	5.34
BD3-LM ($L' = 16$)	Uniform	98.82	4.24	111.22	5.43
Dependency-aware sampler held fixed; only the training-time masking policy differs					
MDLM	Uniform	56.63	4.24	25.74	5.29
MDLM	Dependency-Aware (ours)	**53.33**	4.22	**23.92**	5.22
BD3-LM ($L' = 16$)	Uniform	62.92	4.25	41.12	5.24
BD3-LM ($L' = 16$)	Dependency-Aware (ours)	**57.43**	4.23	**36.99**	5.21

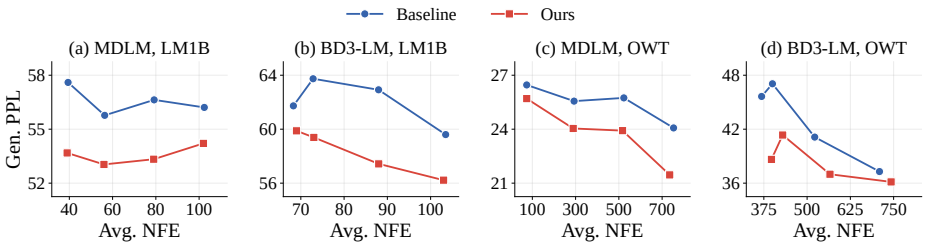


Figure 3: Generative perplexity versus NFE on LM1B and OpenWebText. NFE is varied by sweeping the switching ratio $r^\star$ of the dependency-aware multi-token sampler, applied identically to the uniform-masking baseline and our dependency-aware variant. Dependency-aware masking consistently improves generative perplexity across a wide range of NFE budgets.

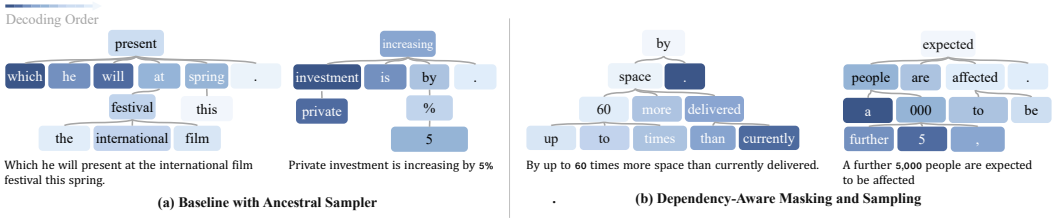


Figure 4: **Dependency trees of two generated sentences,** with each word colored by its unmasking rank (dark blue = first, light blue = last). **Left:** the uniform baseline with the ancestral sampler shows no alignment between unmasking order and tree depth. **Right:** our full method exhibits a leaf-to-root traversal, with the syntactic root unmasked last.

lowest Gen. PPL across all configurations, even the plain ancestral sampler. This indicates that our masking works as a standalone training signal: the custom sampler amplifies the gain, but is not required. Appendix F.1 extends this comparison to multi-token confidence-based samplers, where the training-side gains are largest, and Appendix F.2 to the released DAPD sampler.

5.2 Generation Order Analysis

Training-sampler synergy. Dependency-Aware Masking (Sec. 4.2) biases the forward process so that shallow tokens are masked more often, encouraging the model to reconstruct peripheral modifiers first and the syntactic core last. Whether the trained model actually decodes in this order is an empirical question, which we now address.

Word-level generation-order score. For each generated sentence, we record the unmasking step t_i of every word, parse the sentence with spaCy (Honnibal et al., 2020) to obtain its dependency tree,

Table 2: **The training-side gain is not tied to our sampler: dependency-aware masking helps under every sampler we tested, including two we did not design** (MDLM, LM1B). Each column fixes the inference procedure and varies only the training-time masking policy, so every column is an independent measurement of the training-side effect. *Ancestral* is the original analytical reverse-process sampler of MDLM/BD3-LM and *Entropy* is a single-token sampler that commits one position per step; neither is part of our method. *Deep-First* reverses the depth-aware bias and is reported as a control. Best per column in **bold**.

Training-time masking	Dependency-Aware Sampler (ours)		Structure-agnostic samplers	
	NFE≈40	NFE≈80	Ancestral (NFE=128)	Entropy (NFE=128)
Uniform	57.60	56.63	71.45	55.40
Dependency-Aware, Shallow-First (ours)	**53.68**	**53.33**	**68.42**	**52.98**
Deep-First (control, reversed bias)	57.04	56.57	70.62	54.76

Table 3: **Word-level generation-order score A_w on LM1B** (300 samples, eos-truncated). A_w is the distance-weighted fraction of parent–child edges where the child is unmasked first; $A_w > 0.5$ indicates leaf-to-root decoding. Best score within each training-time masking policy in **bold**.

Training-time masking	Sampler	Order Score A_w
Uniform	Ancestral	0.4979
Uniform	Ours	**0.5311**
Dependency-Aware (ours)	Ancestral	0.5175
Dependency-Aware (ours)	Ours	**0.7095**

and aggregate over parent–child edges (p, c):

$$A_w \;=\; \frac{\sum_{(p,c)} \mathbb{1}[t_c < t_p]\;|t_c - t_p|}{\sum_{(p,c)} |t_c - t_p|}, \tag{3}$$

the fraction of total parent–child time-distance accounted for by edges where the child was unmasked first. $A_w > 0.5$ indicates leaf-to-root (periphery-to-core) decoding; $A_w \approx 0.5$ matches a random schedule and indicates no alignment.

Table 3 reports A_w for the four combinations of {Uniform, Dependency-Aware} training $\times$ {Ancestral, ours} sampler. The leaf-to-root order materializes only when both components are paired: A_w jumps from 0.531 (Uniform + ours) to 0.710 (Dependency-Aware + ours), while under the ancestral sampler A_w stays close to 0.5 for both checkpoints. A structure-agnostic sampler erases the order signal even on a depth-trained model, and a depth-agnostic checkpoint gives our sampler little to align to.

Qualitative comparison. Figure 4 shows two representative samples from the extreme settings of Table 3, confirming the trend visually: the baseline scatters unmasking ranks across the tree, while our full method follows a clear leaf-to-root traversal (more examples in Appendix D).

5.3 Comparison with Alternative Non-Uniform Masking Policies

Setup. To determine if the gain comes from syntactic structure or simply breaking the uniform masking assumption, we compare our dependency-depth signal against three alternative weightings (w_i): **Random Perturbation** (i.i.d. uniform weights), **Linear Positional** (mimicking left-to-right order), and **Frequency-Based** (unigram frequency (Kosmopoulou et al., 2025)). All variants share the exact same training and sampling configurations.

Results. Table 4 shows that Random Perturbation, Linear Positional, and Frequency-Based masking all hover near the uniform baseline, with no clear separation. Dependency-Aware Masking is the only policy that yields a substantial gain (3.3 PPL over Uniform), indicating that the improvement comes from syntactic depth specifically rather than from non-uniformity alone.

5.4 Ablations and Analysis

Strength of structural bias. We sweep the interpolation coefficient λ, defined in Sec. 4.2 and detailed in Appendix A.1, which controls how strongly the depth signal modulates the masking probability. Figure 5 shows that, across both backbones and both sampler settings, Gen. PPL drops sharply as λ increases from zero, reaches a minimum at a small but non-zero value ($\lambda^\star \in \{0.0625, 0.125\}$), and degrades at larger values. The optimum is consistent across all four panels,

Table 4: Comparison of non-uniform masking policies on MDLM (LM1B). All methods share the same reweighting framework; only the per-token weight signal differs.

	Uniform	Random Perturb.	Linear Pos.	Freq.-Based	Dep.-Aware (Ours)
Gen. PPL↓	56.63	55.30	55.51	56.89	**53.33**

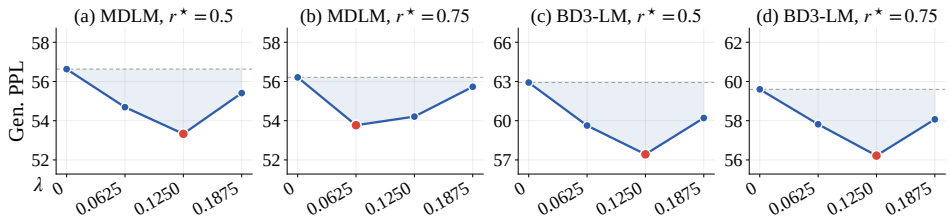


Figure 5: **Effect of interpolation strength λ on LM1B generative perplexity** under our dependency-aware multi-token sampler. $\lambda = 0$ is the uniform-masking baseline (dashed line); the shaded region marks PPL improvements over it. We additionally vary the sampler switching ratio $r^\star \in \{0.5, 0.75\}$ to verify that the optimum is stable across sampler settings.

Table 5: **Parser sensitivity on LM1B (MDLM).** We re-train with depth signals from Stanza instead of our default spaCy parser, and evaluate against the uniform-masking baseline across three samplers. Best within each column in **bold**.

Gen. PPL↓	**Dep.-Aware Sampler (NFE≈80)**	**Ancestral**	**Entropy**
Uniform	56.63	71.45	55.40
Dependency-Aware (Stanza)	**56.11**	**69.01**	**52.93**

indicating that a moderate structural bias is robust to the choice of backbone and to the sampler's switching ratio. Beyond the optimum, over-relying on the depth signal amplifies noise from parser errors and unusually deep trees, hurting generation quality.

Direction of reweighting. We evaluate the direction of depth-based reweighting by comparing two configurations against the uniform baseline: *Deep-First* (deeper tokens masked more) and our default *Shallow-First* (shallower tokens masked more). Table 2 shows Shallow-First consistently outperforms Deep-First across all samplers. This confirms that masking the syntactic core more frequently provides a better training signal, teaching the model to build peripheral context before resolving central tokens.

Sensitivity to parser choice. Finally, we verify our method's robustness to parser choice by re-training MDLM on LM1B using Stanza (Qi et al., 2020) instead of spaCy (Honnibal et al., 2020). Table 5 shows that Stanza-derived masking still outperforms the uniform baseline across all samplers. This indicates our performance gains stem from the coarse tree structure, not fine-grained parsing accuracy.

6 Conclusion and Limitations

In this work, we identify the factorization error in parallel decoding for Masked Diffusion Language Models (MDLMs), where structure-agnostic training and sampling can violate conditional independence by co-decoding strongly coupled tokens. To resolve this, we introduce a non-intrusive Dependency-Aware Framework that aligns generation with natural language hierarchies. Its core is a depth-aware training mask that biases the model toward a periphery-to-core decoding order, which we pair with a graph-based multi-token sampler. Extensive experiments confirm this explicit linguistic strategy consistently improves generation quality and efficiency across various diffusion backbones and inference budgets. Our framework's limitations include a reliance on off-the-shelf parsers that may struggle with noisy text and an evaluation scope restricted to 110M-parameter unconditional generation due to compute constraints. Moreover, since the masking weights depend on the parse of the clean text, which is unavailable at generation time, the unchanged MDLM loss is no longer an exact ELBO of the sampling process.

AI USE STATEMENT

We used a large language model to polish the writing of this paper: it was used to improve the grammar, wording, and readability of text that the authors had already drafted, and all of its suggestions were reviewed and edited by the authors. We did not use generative AI for any of the tasks for which ICLR requires disclosure. In particular, generative AI was not used to propose or refine hypotheses, develop theoretical models or conceptual frameworks, formulate mathematical claims or assist in the writing of proofs, design or provide feedback on our methodology or experiments, implement methods, generate synthetic data, clean or reformat datasets, or interpret results; all of these were carried out by the authors. We take full responsibility for the final content of this work, including all text, claims, and artifacts.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Jiaqi Han, Zhihan Yang, Zhixuan Qi, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

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.

Iskander Azangulov, Teodora Pandeva, Niranjani Prasad, Javier Zazo, and Sushrut Karmalkar. Parallel sampling from masked diffusion models via conditional independence testing. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=XjcHRIu0iF.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=WBcBhT1NKO.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. Llada2. 0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*, 2025.

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.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11315–11325, 2022.

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.

Jiaao Chen, Aston Zhang, Mu Li, Alex Smola, and Diyi Yang. A cheaper and better diffusion language model with soft-masked noise. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4765–4775, 2023.

Lingjie Chen, Ruizhong Qiu, Yuyu Fan, Yanjun Zhao, and Hanghang Tong. Trims: Trajectory-ranked instruction masked supervision for diffusion language models. *arXiv preprint arXiv:2604.00666*, 2026.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

Do Huu Dat, Duc Anh Do, Anh Tuan Luu, and Wray Buntine. Discrete diffusion language model for efficient text summarization. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6293–6305, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.352. URL https://aclanthology.org/2025.findings-naacl.352/.

Marie-Catherine De Marneffe and Christopher D Manning. The stanford typed dependencies representation. In *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, pp. 1–8, 2008.

DeepMind. Gemini diffusion. https://deepmind.google/models/gemini-diffusion/, 2025.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Aaron Gokaslan and Vanya Cohen. 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.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=j1tSLYKwg8.

Shansan Gong, Ruixiang ZHANG, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=58NA3unZj5.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36:16693–16715, 2023.

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuan-Jing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 4521–4534, 2023.

Michael Hersche, Samuel Moor-Smith, Thomas Hofmann, and Abbas Rahimi. Soft-masked diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Gba02UMvrG.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength natural language processing in Python. Zenodo, 2020.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv e-prints*, pp. arXiv–2506, 2025.

BumJun Kim, Dongjae Jeon, Moongyu Jeon, and Albert No. DAPD: Dependency-aware parallel decoding via attention for diffusion LLMs. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=LZro8iqjiA.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M. Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=DjJmre5IkP.

Jaeyeon Kim, Lee Cheuk Kit, Carles Domingo-Enrich, Yilun Du, Sham M. Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Samuel Albergo. Any-order flexible length masked diffusion. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=ttuNnMRI6H.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Hyukhun Koh, Minha Jhang, Dohyung Kim, Sangmook Lee, and Kyomin Jung. Conditional [mask] discrete diffusion language model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8910–8934, 2025.

Despoina Kosmopoulou, Efthymios Georgiou, Vaggelis Dorovatas, Georgios Paraskevopoulos, and Alexandros Potamianos. Masked diffusion language models with frequency-informed training. In *Proceedings of the First BabyLM Workshop*, pp. 531–539, 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.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Anji Liu, Oliver Broadrick, Mathias Niepert, and Guy Van den Broeck. Discrete copula diffusion. *arXiv preprint arXiv:2410.01949*, 2024.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Lizhuo Luo, Zhuoran Shi, Jiajun Luo, Zhi Wang, Shen Ren, Wenya Wang, and Tianwei Zhang. Dawn: Dependency-aware fast inference for diffusion llms. *arXiv preprint arXiv:2602.06953*, 2026.

Linrui Ma, Yufei Cui, Kai Han, and Yunhe Wang. Mask is what dllm needs: A masked data training paradigm for diffusion llms, 2026. URL https://arxiv.org/abs/2603.15803.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*, 2024.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 1659–1666, 2016.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *International Conference on Learning Representations*, 2025.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020. URL https://nlp.stanford.edu/pubs/qi2020stanza.pdf.

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.

Liran Ringel, Ameen Ali, and Yaniv Romano. Dependency-guided parallel decoding in discrete diffusion language models. *arXiv preprint arXiv:2604.02560*, 2026.

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.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=xcqSOfHt4g.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, et al. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*, 2022.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. *arXiv preprint arXiv:2211.16750*, 2022.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=rvZv7sDPV9.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=IJryQAOy0p.

Renzhi Wang, Jing Li, and Piji Li. Infodiffusion: Information entropy aware diffusion process for non-autoregressive text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 13757–13770, 2023.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3Z3Is6hnOT.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Jian Jiao, Juntao Li, Jian Guo, Nan Duan, Weizhu Chen, et al. Ar-diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems*, 36:39957–39974, 2023.

Minkai Xu, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. Energy-based diffusion language models for text generation. *arXiv preprint arXiv:2410.21357*, 2024.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Yiming Zeng, Jinghan Cao, Zexin Li, Yiming Chen, Tao Ren, Zhuochun Li, Dawei Xiang, Xidong Wu, Shangqian Gao, and Tingting Yu. Treediff: Ast-guided code generation with diffusion llms. *arXiv preprint arXiv:2508.01473*, 2025.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation. *arXiv preprint arXiv:2302.05737*, 2023.

Xueyu Zhou, Yangrong Hu, and Jian Huang. Dos: Dependency-oriented sampler for masked diffusion language models. *arXiv preprint arXiv:2603.15340*, 2026.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025a.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, et al. Llada-moe: A sparse moe diffusion language model. *arXiv preprint arXiv:2509.24389*, 2025b.

A Algorithm Details

A.1 Details of Dependency-Aware Masking

Dependency extraction. Given a training dataset, we use an off-the-shelf dependency parser to extract the depth of each word in every sentence as a one-time preprocessing step. Since the language model operates on subword tokens, we align word-level depths to token-level depths: when a word is split into multiple subword tokens by the tokenizer, all resulting tokens inherit the depth of the original word. This produces a depth sequence $\mathbf{d} = (d_1, \ldots, d_N)$ for each tokenized training sample $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, stored alongside the token ids and loaded during training at negligible additional cost.

Forward masking. Algorithm 1 shows the per-batch training procedure with our dependency-aware forward masking. The only departure from standard MDLM is the per-token masking rate $1 - \alpha_t^i$: instead of using the schedule value $1 - \alpha_t$ uniformly across positions, we modulate it with a depth-derived weight w_i, computed in three stages following Section 4.2—a clipped reverse-depth weight $\tilde{w}_i$, an interpolation with the uniform baseline controlled by λ, and a budget-preserving rescaling that fixes the per-sequence mean of w_i to one. Because the rescaling is applied per sequence, the expected number of masked tokens at any t matches the uniform baseline whenever the clipping to $1 - \epsilon$ in Algorithm 1 is inactive,

$$\mathbb{E}\left[\sum_{i=1}^{N} \mathbb{1}\{\mathbf{z}_t^i = \mathbf{m}\}\right] \;=\; (1 - \alpha_t)\, N, \tag{4}$$

which holds except at the highest noise levels, where $(1 - \alpha_t)w_i$ can exceed $1 - \epsilon$ for the most heavily weighted tokens and the expected count falls below $(1 - \alpha_t)N$. No other part of the training pipeline (loss, optimizer, schedule) needs to change. The only added per-step cost is the cheap weight computation in lines 2–4.

Algorithm 1 Dependency-Aware Forward Masking (per training step)

Require: Tokenized sample $x_0 = (x_1, \ldots, x_N)$ with depth sequence $\mathbf{d} = (d_1, \ldots, d_N)$; noise schedule α_t; interpolation strength λ; sharpness γ; clip bounds $p_{\min}, p_{\max}$.
Ensure: Masked input x_t and per-token mask rates $\{1 - \alpha_t^i\}_{i=1}^N$ for the loss.
1: Sample diffusion time $t \sim \mathcal{U}[0, 1]$ and look up base rate $1 - \alpha_t$.
2: $\tilde{w}_i \leftarrow \mathrm{clip}\big(1 - \big(d_i / \max_j d_j\big)^{\gamma},\ p_{\min},\ p_{\max}\big)$ ▷ depth-aware raw weight, shallow > deep
3: $\bar{w}_i \leftarrow \lambda\, \tilde{w}_i + (1 - \lambda) \cdot 1$ ▷ interpolate with uniform baseline
4: $w_i \leftarrow \bar{w}_i \,/\, \frac{1}{N} \sum_{j=1}^{N} \bar{w}_j$ ▷ budget-preserving normalization, $\mathbb{E}_i[w_i] = 1$
5: $p_t^i \leftarrow \min\{1 - \epsilon,\ (1 - \alpha_t)\, w_i\}$ for $i = 1, \ldots, N$
6: $\alpha_t^i \leftarrow 1 - p_t^i$
7: Sample $b_i \sim \mathrm{Bernoulli}(1 - \alpha_t^i)$ independently for each i.
8: $x_t^i \leftarrow$ `[MASK]` if $b_i = 1$ else $x_t^i \leftarrow x_0^i$
9: **return** x_t, $\{1 - \alpha_t^i\}_{i=1}^N$ ▷ used by the standard MDLM denoising loss

Default values. Unless otherwise specified, we use $\gamma = 0.125$, $p_{\min} = 10^{-3}$, and $p_{\max} = 0.95$ across all backbones and datasets.

A.2 Details of Dependency-Aware Multi-Token Sampler

Algorithm 2 provides the full pseudocode of the dependency-aware multi-token sampler described in Section 4.3. Our sampler retains the core mechanism of DAPD (Kim et al., 2026a), namely an interaction graph derived from self-attention from which an independent set of masked positions is unmasked in parallel, and specifies the remaining design choices as in Algorithm 2 and Appendix C. It therefore does not reproduce the released DAPD implementation, which differs in several details (e.g., row-normalized attention scores and layer selection). We expand on each component below.

Interaction graph. At every iteration, the model performs a single forward pass on the current sequence to produce both per-position categorical distributions $p_\theta(\cdot \mid \mathbf{z}_t)$ and the top-layer attention

Algorithm 2 Dependency-Aware Multi-Token Sampler

Require: Denoising network x_θ; sequence length N; switching ratio $r^\star$; edge thresholds $\delta_{\min}, \delta_{\max}$; confidence threshold $c^\star$; per-step cap K; nucleus mass p_{nuc}.
Ensure: Generated sequence x.

Initialize $x \leftarrow (\texttt{[MASK]}, \ldots, \texttt{[MASK]})$ of length N.
while $\mathcal{M} \leftarrow \{i : x^i = \texttt{[MASK]}\} \neq \emptyset$ **do**
  $\{p_\theta(\cdot \mid x)\},\ \{a_{ij}\} \leftarrow x_\theta(x)$ ▷ logits + top-layer attention from one forward pass
  $c_i \leftarrow \max_v p_\theta(x^i = v \mid x), \quad \forall\, i \in \mathcal{M}$
  $r \leftarrow |\mathcal{M}|/N$ ▷ current mask ratio
  if $r \geq r^\star$ **then** ▷ **Graph phase**
    $\delta \leftarrow \delta_{\min} + (1 - r)\dfrac{\delta_{\max} - \delta_{\min}}{1 - r^\star}$
    $s_{ij} \leftarrow \frac{1}{2}(a_{ij} + a_{ji}), \quad \mathcal{E} \leftarrow \{(i,j) \in \mathcal{M}^2 : s_{ij} > \delta\}$
    $\pi \leftarrow$ indices of $\mathcal{M}$ sorted by c_i descending
    $\mathcal{I} \leftarrow \emptyset$
    for $i \in \pi$ **do** ▷ greedy independent set
      if $(i,j) \notin \mathcal{E}$ for all $j \in \mathcal{I}$ **then**
        $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$
      end if
    end for
    $\mathcal{S} \leftarrow$ top-K of $\mathcal{I}$ by c_i
  else ▷ **Confidence phase**
    $\mathcal{C} \leftarrow \{i \in \mathcal{M} : c_i > c^\star\}$
    if $\mathcal{C} = \emptyset$ **then**
      $\mathcal{C} \leftarrow \{\arg\max_{i \in \mathcal{M}}\ c_i\}$ ▷ minimum-one fallback
    end if
    $\mathcal{S} \leftarrow$ top-K of $\mathcal{C}$ by c_i
  end if
  for $i \in \mathcal{S}$ **do** ▷ commit unmasking with nucleus sampling
    $\mathcal{V}_i \leftarrow \textsc{NucleusSet}\big(p_\theta(\cdot \mid x),\ p_{\text{nuc}}\big)$
    $\tilde{p}_\theta(v \mid x) \propto p_\theta(v \mid x)\mathbb{1}[v \in \mathcal{V}_i]$
    $x^i \sim \tilde{p}_\theta(\cdot \mid x)$
  end for
end while
return x

weights $\{a_{ij}\}$. From the latter we form the symmetric interaction score $s_{ij} = \frac{1}{2}(a_{ij} + a_{ji})$ and threshold it to obtain the edge set $E_t = \{(i,j) \in M_t^2 : s_{ij} > \delta_t\}$, where M_t is the current set of masked positions. From the former we obtain the per-position confidence $c_i = \max_v p_\theta(x^i = v \mid \mathbf{z}_t)$. An edge marks a pair of strongly coupled positions that should not be unmasked in the same step. The mask ratio $r_t = |M_t|/N$ then selects between two phases. For BD3-LM, which generates one block at a time, we apply the same procedure within each block: M_t contains only the masked positions of the current block, and r_t is computed relative to the block length.

Graph-based phase. When $r_t \geq r^\star$, we select positions to unmask in parallel by constructing a greedy independent set on $G_t = (M_t, E_t)$. Walking through M_t in descending order of c_i, we admit each position only if it shares no edge with any already-admitted one; the per-step parallel set S_t is then the top-K elements of this independent set ranked by confidence. Confidence as the priority naturally aligns with the periphery-to-core order encouraged during training, since the model assigns higher confidence to peripheral tokens it was trained to reconstruct first. The edge threshold itself is annealed with the mask ratio,

$$\delta_t = \delta_{\min} + (1 - r_t)\,\frac{\delta_{\max} - \delta_{\min}}{1 - r^\star}, \tag{5}$$

rising linearly from $\delta_{\min}$ at the start of decoding ($r_t = 1$) to $\delta_{\max}$ at the phase switch ($r_t = r^\star$), so that fewer pairs are treated as coupled as more context is decoded.

Confidence-threshold phase. When $r_t < r^\star$, the remaining masked positions are concentrated in the densely coupled core, where large-scale parallel decoding is no longer suitable. We therefore switch to a conservative confidence-threshold rule that unmasks only positions clearing a confidence bar $c^\star$, subject to the same per-step cap K. To prevent late-stage stagnation—where all remaining positions may sit below $c^\star$ yet must still be resolved—we augment the rule with a *minimum-one* fallback that force-unmasks the single most confident position whenever the thresholded set is empty:

$$S_t = \text{Top-}K_c\Big(\{i \in M_t : c_i > c^\star\} \cup \{\arg\max_{i \in M_t} c_i\}\Big). \tag{6}$$

The graph phase requires no such safeguard, as its greedy construction always admits the highest-confidence position as its first element, so at least one token is committed per step by construction.

Token sampling and NFE control. For each selected position, we sample its token from the nucleus-truncated and renormalized distribution $\tilde{p}_\theta$ with mass p_{nuc}, which suppresses the low-probability vocabulary tail without altering the relative ranking among plausible candidates. The procedure terminates as soon as no masked positions remain. Because the two phases differ sharply in per-step parallelism, the switching ratio $r^\star$ is the most direct hyperparameter for controlling the total number of function evaluations: lowering $r^\star$ prolongs the parallel-heavy graph phase and reduces NFE, while raising $r^\star$ enters the conservative phase earlier. Other thresholds such as δ_t and $c^\star$ also influence NFE, but only through local density effects within a single phase rather than the qualitative composition of decoding.

B Experimental Details

Datasets We follow the standard setups of MDLM (Sahoo et al., 2024) and BD3-LM (Arriola et al., 2025), and train and evaluate on two language modeling datasets: **LM1B** and **OpenWebText** (OWT). LM1B is tokenized with the `bert-base-uncased` tokenizer at a context length of 128, and OWT is tokenized with the GPT-2 BPE tokenizer at a context length of 1024. For both datasets, we pack tokenized documents into fixed-length sequences using the standard concatenate-and-wrap protocol: documents are concatenated with end-of-text separators and split into non-overlapping chunks of the target context length, with no padding. We use the official train/validation splits for LM1B. For OWT, which has no canonical validation split, we reserve the last 100,000 documents as the held-out validation set.

Dependency annotations. Dependency trees are extracted offline as a one-time preprocessing step, so during the training, they impose no additional computational overhead. Our default parser is spaCy (Honnibal et al., 2020) with the `en_core_web_sm` model, which uses a lightweight transition-based parser, suitable for large-scale preprocessing. For the parser-sensitivity ablation in Table 5, we additionally parse with Stanza (Qi et al., 2020) using the English Universal Dependencies pipeline (`tokenize, mwt, pos, lemma, depparse`), a neural graph-based parser that achieves higher accuracy on standard benchmarks but is substantially slower than spaCy.

Model Architectures and Baselines We apply our method on top of two masked diffusion language modeling backbones, MDLM (Sahoo et al., 2024) and BD3-LM (Arriola et al., 2025), using the released training configurations of each. Both backbones share the same DiT architecture adapted to text with 12 transformer layers, 12 attention heads, and a hidden dimension of 768, which corresponds to 110M parameters.

Training All models are trained for 1,000K steps with a global batch size of 512, a learning rate of 3×10^{-4} with a linear warmup over the first 2,500 steps. For **BD3-LM**, we follow the two-stage training protocol of Arriola et al. (2025), first training an MDLM model on the same data for 850K steps, then initializing the BD3-LM model with the MDLM weights and training for an additional 150K steps. For **MDLM**, we train a single stage for 1,000K steps. We use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and zero weight decay. Gradients are clipped at global norm 1.0, and training is performed in bf16 mixed precision. We maintain an exponential moving average (EMA) of the model weights with decay 0.9999, and use the EMA weights for all evaluations. All settings follow the released configurations of MDLM and BD3-LM exactly.

Samplers Unless otherwise specified, all experiments use our **dependency-aware multi-token sampler** described in Section 4.3 and Appendix A.2. For the robustness study in Table 2 and the parser-sensitivity study in Table 5, we additionally evaluate two structure-agnostic baseline samplers, described below. All baselines operate on the same trained MDLM / BD3-LM checkpoint as our sampler and differ only in how they select which masked positions to commit at each step.

Ancestral sampler. The original sampler shipped with MDLM (Sahoo et al., 2024) and BD3-LM (Arriola et al., 2025), which corresponds to the analytical reverse process of the absorbing diffusion. At each step it draws the unmasking pattern according to the schedule-implied transition kernel, independently of model confidence. We use the released implementations without modification.

Entropy-based single-token sampler. A greedy decoder that, at each step, runs one forward pass, computes per-position predictive entropy $H_i = -\sum_v p_\theta(x^i = v \mid \mathbf{z}_t) \log p_\theta(x^i = v \mid \mathbf{z}_t)$ over masked positions, commits the single lowest-entropy position via nucleus sampling with mass p_{nuc}, and repeats until no masked positions remain. The total number of denoising steps therefore equals the number of masked positions in the initial sequence (one token per step).

Compute & Efficiency All experiments are run on NVIDIA H100 GPUs. Our method introduces negligible training overhead beyond the MDLM and BD3-LM baselines: dependency-aware masking only modifies the per-token masking probabilities sampled by the dataloader, and dependency trees are extracted once offline as a preprocessing step. At inference, the multi-token sampler requires no additional model evaluations, adding only modest per-step overhead for attention extraction and graph construction (Appendix F.3).

C Evaluation Details

Generative Quality. We evaluate generation quality using **Generative Perplexity** (Gen. PPL, lower is better), computed by scoring unconditionally generated samples under a pretrained GPT-2-Large evaluator (Radford et al., 2019). For each model and decoding configuration, we draw 300 samples for LM1B and 100 samples for OWT, and report the mean Gen. PPL across samples. For LM1B, samples have a fixed length of 128 tokens, matching the training context. For OWT, where samples reach the GPT-2-Large context window of 1024 tokens, we follow the protocol of Sahoo et al. (2024) and compute Gen. PPL using a sliding window with stride 512, averaging log-likelihoods over all overlapping windows. To avoid reward-hacking the metric via low-entropy decoding (Wang et al., 2026), we fix nucleus sampling at $p_{\text{nuc}} = 0.9$ for all baselines and our method, and we additionally report the per-sample token-level entropy alongside Gen. PPL where relevant.

Sampler hyperparameters. Our dependency-aware multi-token sampler exposes six hyperparameters: switching ratio $r^\star$, edge thresholds $\delta_{\min}$ and $\delta_{\max}$, confidence threshold $c^\star$, per-step cap K, and nucleus mass p_{nuc}. Because MDLM denoises the full sequence in parallel while BD3-LM denoises one block at a time, and because LM1B (length 128) and OWT (length 1024) differ by an order of magnitude in sequence length, we tune separate defaults for each (backbone, dataset) combination. We fix $p_{\text{nuc}} = 0.9$, $r^\star = 0.5$, and $c^\star = 0.9$ across all settings. For **MDLM**, we use $\delta_{\min} = 0.003$, $\delta_{\max} = 0.005$, and $K = 4$ on LM1B; on OWT we keep $\delta_{\min}$ and $\delta_{\max}$ unchanged but raise $K = 16$ to exploit the longer sequence. For **BD3-LM** with block size $L' = 16$, we use $\delta_{\min} = 0.01$, $\delta_{\max} = 0.05$, $K = 2$ on LM1B, and $K = 4$ on OWT. The smaller K for BD3-LM reflects its shorter intra-block denoising window. Top-layer attention is averaged across the last two transformer layers and all heads to obtain the symmetric interaction score s_{ij}. Baseline samplers follow their original specifications: the ancestral sampler uses the schedule shipped with MDLM (Sahoo et al., 2024) and BD3-LM (Arriola et al., 2025), and the entropy-based single-token sampler commits one position per step under nucleus sampling with $p_{\text{nuc}} = 0.9$.

D Qualitative Samples

Unmasking trajectory: ancestral vs. ours. Figure 6 complements the quantitative generation-order analysis in Section 5.2: while A_w measures alignment with dependency depth, this visualization shows the sequence-level effect. Two mechanisms in our sampler shape the trajectory. First, the graph-phase independent-set rule forces the K commits within a single step to be non-adjacent

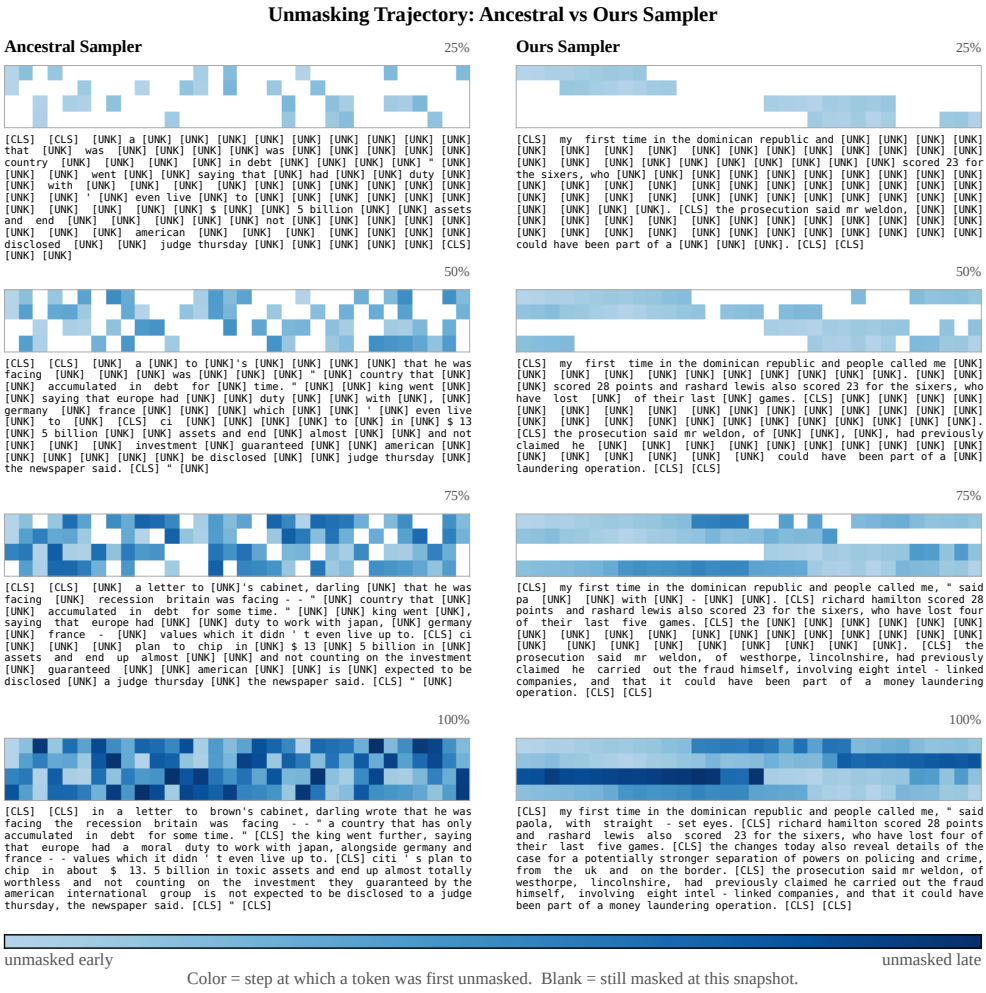


Figure 6: Unmasking trajectories of the ancestral sampler (left) and our dependency-aware multi-token sampler (right) on the same MDLM checkpoint, shown at 25%, 50%, 75%, and 100% of the denoising steps. Cells are colored by the step at which a token was first unmasked (darker = later); blank cells are still masked. Our sampler grows several local clusters in parallel while the ancestral sampler scatters commits uniformly across the sequence.

in the interaction graph, so they tend to lie on different subtrees of the dependency tree, and each step deposits seeds across multiple regions at once. Second, the confidence-phase preference for high-probability positions tends to extend existing clusters, since tokens adjacent to already-decoded context are easier to predict. Together they produce the parallel-region growth visible on the right: at 25% and 50% several disjoint spans are partially formed, and by 75% each cluster is nearly complete, in sharp contrast to the spatially uncorrelated commits of the ancestral sampler on the left.

Generated samples on LM1B. We show a pair of unconditional samples drawn from the same MDLM backbone trained on LM1B, comparing the uniform-masking baseline ($\lambda = 0$) and our dependency-aware variant ($\lambda = 0.125$). Both checkpoints are decoded with the dependency-aware multi-token sampler under identical hyperparameters. Each sample is a 128-token packed sequence following the LM1B convention; document boundaries are denoted by the literal `[CLS]` marker, and the per-sample generative perplexity (Gen. PPL, lower is better) under GPT-2 Large is reported in each box.

Standard MDLM (Uniform), Gen. PPL = 44.23

```
[CLS] taylor told reuters at the end of the meeting.  [CLS] he was
joined by dame joan ruddock, john kufuor and swedish prime minister
stoltenberg, as he was welcomed by south africa ' s third highest
scorer, victor matfield.  [CLS] the two sides have both " to believe
that obama and mccain will have much more influence, on the russian
side of the world and of all - important russia particularly, " said
ken kusch, a senior policy analyst at the center for strategic and
international studies.  [CLS] the thunder, led by paul millsap, had 17
points and 15 rebounds against the nets.  [CLS] in the past [CLS]
```

Ours (Dependency-Aware, $\lambda = 0.125$), Gen. PPL = 40.79

```
[CLS] investigation by the state attorney general.  [CLS] it ' s
been her second year since marrying doug benson of illinois at the
age of 38.  [CLS] nato provided no information on the timings of
the incidents, a defence ministry spokesman said.  [CLS] with a 10
- year - old, it seems almost the best way to get the kids to know
that their mother is okay and to know more as to why she has left the
family.  [CLS] but among other things, the plan would call for more
transparency, increase tarp financing, restore tarp funds, and put the
fed in power, as what obama called the " special economic institution
" [CLS]
```

Figure 7: Generated sample pair from MDLM trained on LM1B (sequence length 128).

E Analysis of the Attention-Derived Interaction Graph

E.1 Alignment with Syntactic Dependencies

Our multi-token sampler (Sec. 4.3) assumes that the attention-derived interaction score s_{ij} identifies pairs of tokens that should not be decoded in the same step. We test this assumption directly against parser-derived syntactic structure.

Setup. We sample 900 validation sentences from LM1B and parse them with spaCy to obtain parser-derived parent–child dependency edges, forming a binary matrix Y with $Y_{ij} = 1$ when the words containing subword tokens i and j are directly connected by a dependency edge. We mask each sequence at ratio $r = 0.5$, run a forward pass of the MDLM backbone, and compute the symmetric interaction score s_{ij} over masked positions exactly as our sampler does. We then measure how well s_{ij} ranks the true dependency edges. Since only 10.6% of candidate pairs are true edges, we report average precision (AP) alongside AUROC; the AP of a random ranking equals this positive rate.

Table 6: Alignment between the attention-derived interaction score s_{ij} and parser-derived dependency edges on 900 LM1B validation sentences (MDLM, masking ratio $r = 0.5$).

Training-time masking	**AUROC (%)↑**	**AP (%)↑**
Random chance	50.0	10.6
Uniform (baseline)	86.7	50.2
Dependency-Aware (ours)	**87.1**	**53.0**

Results. Table 6 shows that both checkpoints rank syntactic edges far above chance, which validates the premise of the sampler: the interaction graph it builds is genuinely syntactic rather than an arbitrary attention threshold. Notably, the uniform-masking baseline already exhibits this alignment, indicating that the structural signal is learned natively by masked diffusion training and is extracted, not created, by our sampler. Dependency-aware masking further sharpens the alignment (AP $50.2 \rightarrow 53.0$), consistent with its role of aligning the training distribution with the decoding trajectory the sampler follows.

E.2 Co-Decoding Attention-Linked Pairs Causes Factorization Errors

Section E.1 shows that s_{ij} aligns with syntactic dependencies. We now test whether the pairs it flags are actually harmful to decode jointly, via a counterfactual intervention on the sampler.

Setup. Using the dependency-aware MDLM checkpoint on LM1B, we compare our sampler, which in its graph phase prevents pairs with $s_{ij} > \delta$ from being unmasked in the same step, against a *Forced-Edge* sampler that deliberately unmasks such pairs together. We define the collision rate as the percentage of cross-word token pairs decoded in the same step whose words are directly connected in the dependency tree,

$$\mathrm{CR} = \frac{\sum_{(i,j)\in\mathcal{C}} Y_{ij}}{|\mathcal{C}|} \times 100, \tag{7}$$

where $\mathcal{C}$ is the set of cross-word token pairs decoded in the same step and Y_{ij} is defined as in Section E.1; pairs of tokens belonging to the same word are excluded.

Table 7: Counterfactual intervention on the sampler (MDLM, LM1B). Forcing attention-linked pairs to be decoded in the same step sharply increases dependency collisions and degrades generation quality.

Sampler	Collision Rate (%)↓	Gen. PPL↓
Forced-Edge (forces high-attention pairs together)	28.06	222.09
Dependency-Aware (ours; blocks high-attention pairs)	**0.50**	**53.33**

Results. As shown in Table 7, forcing attention-linked pairs to be decoded together raises the collision rate from 0.50% to 28.06% and degrades Gen. PPL from 53.33 to 222.09. The edges identified by s_{ij} are therefore not merely correlated with syntax: forcing them to be decoded jointly sharply degrades generation, consistent with the factorization error our sampler is designed to avoid.

F Sampler Comparisons and Inference Cost

F.1 Parallel Decoding with Confidence-Based Samplers

Setup. Confidence-based samplers rank masked positions by a per-token score, such as the top-probability margin, the maximum probability, or the predictive entropy (Chang et al., 2022; Kim et al., 2025), and are often run with one token committed per step, where co-decoding collisions cannot occur. We extend each of them to multi-token parallel decoding by committing the top-k masked positions under its score at every step, at one half and one quarter of the sequential step budget on LM1B. For each sampler, we decode the uniform-masking and the dependency-aware MDLM checkpoints with identical settings, so that each row of Table 8 isolates the effect of the training-time masking policy under a fixed inference procedure.

Table 8: Gen. PPL↓ under multi-token parallel decoding on LM1B (MDLM). Each row fixes the sampler and budget and varies only the training-time masking policy; the NFE column reports the number of function evaluations used by each sampler, so budgets are matched only approximately.

			Training-time masking	
Budget	**Sampler**	**NFE**	**Uniform**	**Dependency-Aware (ours)**
$\approx 1/2$	Dependency-Aware (ours)	56	55.77	**53.04**
	Margin top-k	64	120.59	**101.65**
	Confidence top-k	64	139.73	**111.30**
	Entropy top-k	64	140.97	**127.77**
$\approx 1/4$	Dependency-Aware (ours)	39	57.60	**53.68**
	Margin top-k	32	213.31	**127.64**
	Confidence top-k	32	253.10	**142.87**
	Entropy top-k	32	251.28	**174.35**

Results. Dependency-aware masking lowers Gen. PPL under every sampler and budget in Table 8. The gains are largest exactly where parallel decoding is most aggressive and the sampler offers no structural protection: at the quarter budget, dependency-aware training reduces Gen. PPL from 213.31 to 127.64 for margin top-k, from 253.10 to 142.87 for confidence top-k, and from 251.28 to 174.35 for entropy top-k, whereas under our dependency-aware sampler, which already blocks most co-decoding collisions, the change is from 57.60 to 53.68. This indicates that the structural prior learned during training makes the model itself more robust to collision-prone decoding, independently of the inference procedure.

F.2 Comparison with the Released DAPD Sampler

Setup. We compare our sampler with the released implementation of DAPD (Kim et al., 2026a), decoding the uniform-masking and the dependency-aware MDLM checkpoints on LM1B at a matched budget of NFE≈80. Our sampler differs from the released implementation in the two design choices of Section 4.3, as well as in the implementation details noted in Appendix A.2.

Table 9: Gen. PPL↓ of the released DAPD sampler and our sampler on LM1B (MDLM) at NFE≈80. Each row fixes the sampler and varies only the training-time masking policy. Best per column in **bold**.

	Training-time masking	
Sampler	**Uniform**	**Dependency-Aware (ours)**
DAPD (released implementation)	59.26	57.20
Ours	**56.63**	**53.33**

Results. Dependency-aware masking also improves Gen. PPL under the released DAPD sampler ($59.26 \rightarrow 57.20$), providing further evidence that the training-side gain does not depend on our sampler. Our sampler lowers Gen. PPL under both training policies, and the gain from dependency-aware masking grows from 2.06 under DAPD to 3.30 under our sampler.

This is consistent with the rationale behind the two design choices of Section 4.3. First, degree-weighted priority favors positions with many high-attention edges. Since the interaction graph aligns with syntactic dependencies (Appendix E.1), these are largely the core tokens that dependency-aware masking trains the model to resolve last, so committing them early works against the periphery-to-core order the model has learned. Ranking by confidence alone instead follows the model's own ordering, which favors the peripheral tokens it was trained to reconstruct first (Table 3). Second, without a cap, the independent set can occasionally become large enough to commit many tokens in a single step. Because the graph retains only the strongest couplings, the weaker dependencies among these tokens are then all ignored at once; bounding the number of commits per step limits how much factorization error a single step can introduce.

F.3 Wall-Clock Cost

Preprocessing. Dependency parsing is a one-time offline step. Using 112 CPU cores, parsing takes approximately 32 minutes for LM1B and 6.3 hours for OpenWebText. The parser is not invoked during training steps or at inference.

Inference. We measure end-to-end generation time on LM1B, including attention extraction and graph construction, and compare our sampler with the ancestral sampler of each backbone. Table 10 reports the median generation time, the time per function evaluation, and the speedup relative to the ancestral sampler. The *graph at every step* setting applies the graph phase throughout decoding and is therefore the most expensive configuration of our sampler.

Table 10: End-to-end generation time on LM1B, including attention extraction and graph construction. Speedup is relative to the ancestral sampler of the same backbone.

Backbone	Sampler	NFE	Median time	Time / NFE	Speedup
MDLM	Ancestral	127	3.14 s	24.7 ms	$1.00\times$
	Ours, graph at every step	40	1.08 s	27.0 ms	$\mathbf{2.91}\times$
	Ours, representative setting	79	1.98 s	25.1 ms	$1.59\times$
BD3-LM	Ancestral	127	2.13 s	16.8 ms	$1.00\times$
	Ours, graph at every step	67	1.23 s	18.4 ms	$\mathbf{1.73}\times$
	Ours, representative setting	88	1.58 s	18.0 ms	$1.35\times$

Results. Even when the graph is constructed at every step, the time per function evaluation increases by only about 9% over the ancestral sampler ($24.7 \rightarrow 27.0$ ms for MDLM and $16.8 \rightarrow 18.4$ ms for BD3-LM), while the reduction in NFE yields a net 1.35–$2.91\times$ speedup. The additional cost of graph construction therefore does not offset the savings from fewer decoding steps.