MANTA-LM: PARALLEL TEXT GENERATION WITH LOCALLY STRUCTURED LATENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most text is generated around text that already exists, such as a prompt, a source sentence, or the ending of a passage. Compressed latent spaces make parallel generation cheaper, but encoders that compress a whole passage mix every token into every latent, so known text cannot be held fixed. Manta-LM instead generates in *locally structured latents*: a convolutional TextVAE compresses text so that each latent position summarizes a short span of tokens. The latents of known text stay fixed while a Transformer flow generates the rest, and a decoder reads out all tokens in parallel; changing which latents are fixed switches between generation, continuation, and infilling. At $4\times$ compression, TextVAE reconstructs 99.75% of tokens on average, and joining the latents of two separately encoded texts adds only 0.45 percentage points of token error. With one 101M flow per task, Manta-LM obtains the highest BLEU among the compared models on four conditional generation tasks. Choosing guidance for each step budget lets four sampling steps match 64 default-guidance steps on question generation, and 16 steps reach an unconditional Gen PPL of 46.9, against 78.3 for RADD at similar unigram entropy.

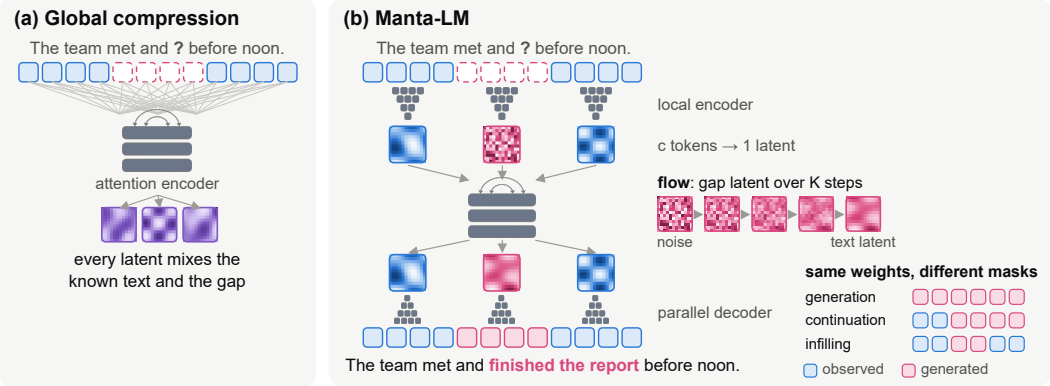


Figure 1: **Manta-LM keeps known text fixed in a compressed latent space.** (a) An encoder that compresses the passage into a few latents mixes every token into every latent, so no latent holds only the known text. (b) The convolutional codec of Manta-LM maps each group of c neighboring tokens to one latent. Latents of observed text stay fixed, latents of the gap start from noise, and a Transformer flow updates the free latents for K steps before a parallel decoder reads out the text. Changing the mask switches between generation, continuation, and infilling. Schematic example; the codec compresses by $c \in \{1, 4\}$.

1 INTRODUCTION

Language models are often asked to write around text that already exists: to continue a prompt, rewrite a source sentence, or fill the gap between two finished paragraphs. Autoregressive models write left to right and use a suffix only after fill-in-the-middle training (Bavarian et al., 2022). Diffusion language models update all positions in parallel and can condition on any of them (Austin et al., 2021; Sahoo et al., 2024; Nie et al., 2025), but every update processes the whole sequence.

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.

Generating in a compressed latent space shortens that sequence, as latent diffusion does for images (Rombach et al., 2022). For text, however, compression and conditioning pull in opposite directions. Encoders that compress a passage into a fixed set of latents mix the whole passage into every latent (Meshchaninov et al., 2025; Lovelace et al., 2023), so no latent corresponds to a span that could be held fixed or replaced. Token-aligned latents keep this correspondence but do not shorten the sequence (Jiang et al., 2026; Hu et al., 2026). We ask whether text can be compressed into *locally structured latents*, in which each latent position summarizes a bounded span of tokens, and whether a generator can use this structure to condition on any observed spans.

Manta-LM answers both questions with two components that divide the work (Figure 1b). A convolutional TextVAE compresses text with local receptive fields, so each latent position summarizes a few neighboring tokens; at $4\times$ compression, 128 tokens become 32 latent positions. The latents of observed text are encoded once and held fixed, and the latents to be generated start from Gaussian noise. A Transformer flow attends across all positions and updates only these free latents; after the last step, a convolutional decoder reads out every token in parallel. The codec only has to preserve local content, which convolutions do almost losslessly, and the flow supplies the long-range dependencies. Changing which positions are observed turns the same model into an unconditional generator, a continuation model, or an infilling model.

The experiments support both parts of this design. TextVAE reconstructs 99.75% of tokens across 19 datasets at $4\times$ compression, and joining the first latent halves of two separately encoded texts costs 0.45 percentage points of token accuracy at $4\times$ and nothing at $1\times$. With one 101M flow per task, Manta-LM obtains the highest BLEU among the compared models on four conditional generation benchmarks. Few sampling steps suffice when guidance matches the step budget: on question generation, four steps reach BLEU 0.180, against 0.173 for 64 steps with the default guidance, using 4 field evaluations instead of 128. Our contributions are:

- TextVAE, a convolutional codec that compresses text into locally structured latents, so latents from different texts can be joined and decoded (Sections 3.1 and 4.3).
- A masked conditional flow that holds the observed latents fixed and generates the free ones jointly, so one model supports generation, continuation, and infilling (Sections 3.2–3.4).
- Experiments on conditional generation, language modeling, infilling, and sampling budgets, which show that guidance should be chosen with the step budget (Section 4).

2 Related work

Autoregressive and iterative generation. Autoregressive models factorize text into token conditionals (Brown et al., 2020; Touvron et al., 2023); fill-in-the-middle training incorporates suffix context (Bavarian et al., 2022), and blank-infilling models insert text between given spans (Donahue et al., 2020; Shen et al., 2020). Discrete diffusion updates token states in parallel (Austin et al., 2021; Lou et al., 2024; Sahoo et al., 2024; Ou et al., 2025; Nie et al., 2025), while block diffusion combines autoregressive context and within-block updates (Arriola et al., 2025). Manta-LM instead updates all free positions jointly in a learned, compressed latent space and then decodes the text in parallel.

Continuous text representations. Continuous language generation operates on embeddings, encoded sequences, or categorical relaxations (Li et al., 2022; Dieleman et al., 2022; Gong et al., 2023; Yuan et al., 2024; Cheng et al., 2024; Jo & Hwang, 2025). Latent diffusion models for text compress a passage into a fixed set of latents with a pretrained encoder (Lovelace et al., 2023; Zhang et al., 2023), and COSMOS learns globally compressed latents (Meshchaninov et al., 2025); in these models a latent can depend on any part of the passage. TextLDM uses a token-aligned Transformer VAE with representation alignment, conditional flow, and parallel decoding (Jiang et al., 2026), which keeps positions aligned but does not shorten the sequence. Manta-LM compresses locally, so observed and generated latents can be assembled by position, and leaves global interaction to the flow.

Generation over compressed or embedded states. Cola-DLM uses a block-causal latent flow (Guo et al., 2026), and CALM generates context-free compressed chunks autoregressively (Shao et al., 2025). ELF and LangFlow learn flows over token embeddings (Hu et al., 2026; Chen et al., 2026). Manta-LM combines local compression with generation over the whole sequence. Table 1 summarizes how these models represent text and how they condition on it.

Table 1: How continuous text generators represent and condition on text, as formulated in each work.

Method	Latent sequence	Generator	Conditioning
COSMOS	compressed by a global encoder	latent diffusion	encoded source
TextLDM	one latent per token	flow	clean prefix
Cola-DLM	one latent per token, causal	block-causal flow	prefix
CALM	one latent per chunk	autoregressive	previous chunks
ELF, LangFlow	token embeddings	flow	prefix or none
Manta-LM	compressed by a local codec	masked global flow	any observed spans

3 MANTA-LM

Figure 2 shows the three stages of Manta-LM: TextVAE maps tokens to latents (A), a flow is trained on the free latents (B), and sampling runs K steps with the observed latents fixed (C). Given tokens $x = (x_1, \ldots, x_L)$, an observation mask splits the positions into observed positions C and free positions F, and the goal is to model $p(x_F \mid x_C, C, F)$. The codec defines the representation, and the flow learns the distribution of the free latents.

3.1 LOCALLY STRUCTURED TEXTVAE

TextVAE shortens the sequence that the flow must process while keeping each region of text recoverable on its own. It adapts deep compression autoencoders for images (Chen et al., 2025) to token sequences, with symmetric, fully parallel encoder and decoder networks. For a packed length L' and compression factor c, it produces $n = L'/c$ latent positions, so 128 tokens become 32 positions at $4\times$ compression; at $1\times$, the length is unchanged.

Both networks use local convolutions and no self-attention. Each latent therefore depends only on a short window of neighboring tokens, so it can be moved into another text without carrying information about the rest of its passage. This divides the work: the codec preserves the content of each region, and the flow learns the dependencies between regions. Section 4.3 tests this property directly by joining the latents of different texts (Figure 5).

The encoder defines a diagonal Gaussian posterior,

$$q_\phi(z_0 \mid x) = \mathcal{N}\big(\mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x))\big)\,. \tag{1}$$

The decoder projects token embeddings through a shared vocabulary matrix with learned scale and bias. Position-wise softmax defines normalized categorical distributions $p_\psi(x \mid z_0)$, without conditioning on previously emitted tokens.

Training combines reconstruction, posterior regularization, and feature stability:

$$\mathcal{L}_{\mathrm{VAE}} = \mathcal{L}_{\mathrm{rec}} + \beta(s) D_{\mathrm{KL}}(q_\phi(z_0 \mid x) \,\|\, \mathcal{N}(0, I)) + \lambda_{\mathrm{stab}} \mathcal{L}_{\mathrm{stab}}, \tag{2}$$

where s is training progress. $\mathcal{L}_{\mathrm{rec}}$ is a binary cross-entropy on the vocabulary logits in which non-padding tokens receive ten times the weight of padding; the categorical likelihood above is used for evaluation. The KL weight $\beta(s)$ starts at 10^{-4} and rises linearly to 10^{-2} over one third of training. $\mathcal{L}_{\mathrm{stab}}$ penalizes intermediate activations whose magnitude exceeds 100 ($\lambda_{\mathrm{stab}} = 10^{-4}$), which keeps training numerically stable. Appendix A gives the architecture and loss reductions.

3.2 CONDITIONING AND PACKING

The observation mask gives each latent one of two roles. In infilling, for example, the latents of the known beginning and ending are observed and stay fixed as h_C, while the latents of the middle are free and evolve as z_F. At $4\times$ compression each latent covers four tokens, so observed and free spans are padded to multiples of four and no latent mixes observed and free tokens. Separate learned projections tell the flow which role each latent plays before global attention. Sampling encodes the observed text once and initializes the free latents from noise.

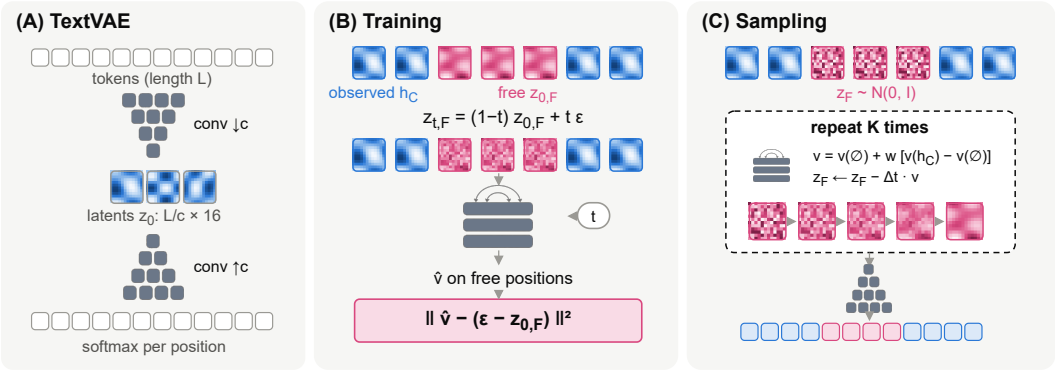


Figure 2: **Architecture.** (A) The TextVAE encodes L tokens with local convolutions into L/c latent positions and decodes every position in parallel with a per-position softmax. (B) Training noises only the free latents and regresses the velocity $\epsilon - z_{0,F}$ at free positions; the observed latents h_C enter as clean conditioning, and time t enters through adaptive normalization. (C) Sampling starts the free latents from noise, applies K guided Euler updates with the observed latents fixed, and decodes the assembled sequence. Blue tiles are observed latents, rose tiles free latents; speckle denotes noise.

3.3 Global conditional flow

The flow supplies the global interaction that the local codec leaves out. The velocity of the free latents depends on the fixed observed latents and on the current values of all free latents:

$$\frac{dz_F(t)}{dt} = v_\theta(z_F(t), h_C, t)\,, \tag{3}$$

where t denotes generation time and h_C remains fixed during sampling.

We train the flow with conditional flow matching (Lipman et al., 2023), interpolating between the clean free latents $z_{0,F}$ and independent Gaussian noise ϵ_F:

$$z_{t,F} = (1-t)z_{0,F} + t\epsilon_F, \qquad u_F = \epsilon_F - z_{0,F}. \tag{4}$$

For a sampled time $t \in [0, 1]$, the regression objective is

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{x,C,F,t,\epsilon}\left[\|v_\theta(z_{t,F}, h_C, t) - u_F\|_F^2\right]. \tag{5}$$

The loss covers only the free positions, and the observed latents enter only as conditioning (Figure 2B). The field predicts the velocity from clean latents toward noise, and sampling follows it backward in time. The field is a Transformer that attends across the whole packed sequence and receives the time through adaptive normalization (Peebles & Xie, 2023) (Figure 2C).

Training data. Each conditional task uses its own flow, trained on that task's source–target pairs with the source observed and the target free. The unconditional flows are trained on general text with no observations. The codec is trained once and kept frozen during flow training.

3.4 Sampling and task modes

Sampling starts the free latents from Gaussian noise and integrates from $t = 1$ to $t = 0$, feeding each updated state back to the field while the observed latents stay fixed. Classifier-free guidance (Ho & Salimans, 2021) combines the conditional and unconditional fields:

$$v_w(z_t, h_C, t) = v_\theta(z_t, \varnothing, t) + w\,[v_\theta(z_t, h_C, t) - v_\theta(z_t, \varnothing, t)]\,. \tag{6}$$

Here $w = 1$ recovers the conditional field. Each sampling step evaluates the field once over the full latent sequence (twice when $w \neq 1$), so the cost of a sample grows with both the number of steps and the length of the latent sequence.

Generation, continuation, and infilling differ only in what is observed: nothing, a prefix, or both a prefix and a suffix. After the last step, the free and observed latents are assembled, unnormalized, and decoded in parallel, and the generated spans are extracted with padding removed. The mask alone adapts one model to these observation patterns (Appendix A.4).

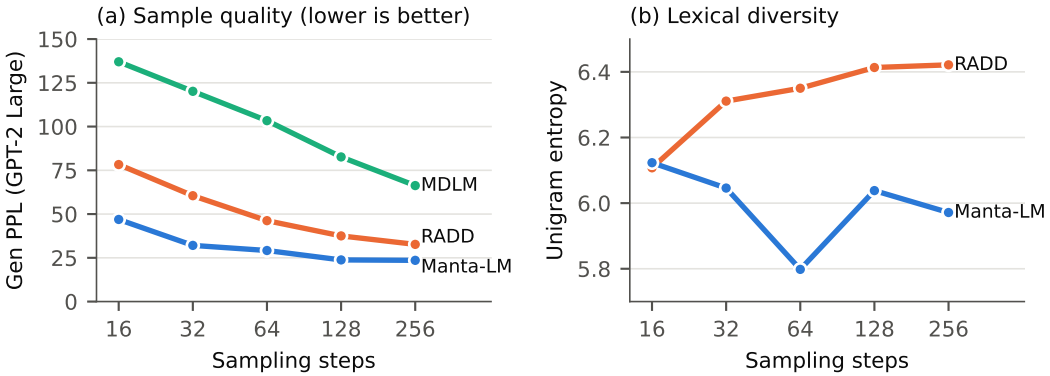


Figure 3: Unconditional generation across sampling budgets. (a) Gen PPL under GPT-2 Large; lower is better. (b) Unigram entropy of the samples. At 16 steps Manta-LM has lower Gen PPL than RADD at similar entropy. Full values in Table 11.

4 Experiments

4.1 Evaluation setup

Models. We use three kinds of flows, all with $1\times$ codecs. Task flows (101M) are trained for each conditional task with a BERT-tokenizer codec. An unconditional 101M flow is trained on OpenWebText with a GPT-2-tokenizer codec and is used for unconditional generation and language modeling. A 0.6B flow (556M parameters) trained on SlimPajama with a GPT-2-tokenizer codec is used for infilling. The $4\times$ codec is evaluated for reconstruction, joining, and throughput. Model sizes count flow parameters; Table 6 lists complete pipelines.

Tasks and metrics. Conditional tasks comprise QQP paraphrasing, Quasar-T question generation, Wiki-Auto simplification, and CCD dialogue (DataCanary et al., 2017; Dhingra et al., 2017; Jiang et al., 2020; Zhou et al., 2018), evaluated by BLEU, ROUGE-L, BERTScore, and Distinct-1. Unconditional samples are scored by generation perplexity (Gen PPL) under GPT-2 Large and by unigram entropy. Language modeling reports an ELBO-based perplexity bound on LAMBADA, WikiText-2/103, PTB, and 1BW.

4.2 Generation quality

Unconditional samples reach lower Gen PPL at every budget. Manta-LM has lower Gen PPL than MDLM and RADD at every budget in Figure 3 (Sahoo et al., 2024; Ou et al., 2025). At 16 steps, its Gen PPL is 46.9 versus 78.3 for RADD at almost the same unigram entropy (6.12 versus 6.11). More steps lower Gen PPL further, to 29.2 at 64 steps and 23.6 at 256 steps.

One small flow per task obtains the highest BLEU. With one 101M flow per task, Manta-LM obtains the highest BLEU among the compared models on all four tasks (Table 2). On paraphrasing it reaches 0.301, against 0.243 for SeqDiffuSeq and 0.206 for fine-tuned GPT-2 large; on simplification it reaches 0.389, against 0.371 for SeqDiffuSeq, and it also has the highest BERTScore on these two tasks. Figure 4 shows how one paraphrase forms: the flow revises all positions in parallel, and words settle in no fixed order, with “way” fixed first and “be” last.

Perplexity is the lowest on all five benchmarks. Manta-LM has the lowest reported perplexity on all five datasets in Table 3, including a GPT-2-small baseline retrained for 80,000 updates with the same tokenizer, data, and schedule. For Manta-LM we report $\exp(-\mathcal{L}_{\mathrm{ELBO}}/L)$, an upper bound on perplexity in which the flow prior density is evaluated by the change of variables (Appendix C).

Table 2: Conditional text generation. BLEU and BERTScore (BS), higher is better; bold marks each column's best result. Manta-LM uses one 101M flow per task. Full ROUGE-L, diversity, and parameter-count results are in Appendix D. Baselines: fine-tuned (FT) GPT-2 (Radford et al., 2019), GPVAE-T5 (Du et al., 2022), LevT (Gu et al., 2019), DiffuSeq (Gong et al., 2023), SeqDiffuSeq (Yuan et al., 2024), and RDLM (Jo & Hwang, 2025).

	Paraphrase		Question		Simplification		Dialogue	
Model	BLEU	BS	BLEU	BS	BLEU	BS	BLEU	BS
GPT-2 base FT	.198	.825	.074	.605	.308	.802	.011	.528
GPT-2 large FT	.206	.836	.111	**.635**	.269	.788	.013	**.529**
GPVAE-T5	.241	.847	.125	.631	.339	.817	.011	.432
LevT	.227	.834	.093	.549	.205	.725	.016	.476
DiffuSeq	.241	.837	.173	.612	.362	.813	.014	.513
SeqDiffuSeq	.243	.840	.175	.617	.371	.821	.011	.443
RDLM	.215	.704	.179	.576	.280	.687	.011	.481
Manta-LM	**.301**	**.848**	**.186**	.613	**.389**	**.822**	**.020**	.485



Figure 4: Intermediate decodings of the QQP flow for one input, at selected steps from noise (top) to the final output (bottom). Rose boxes mark words that match the final paraphrase.

4.3 Locally structured latents

Compression preserves text content. At $4\times$ compression, TextVAE reconstructs 99.75% of tokens on average across 19 datasets covering general text, code, and mathematics, and at least 99.35% on every dataset, so the compressed latents retain almost all of the text.

Joined latents decode to joined text. We encode two texts separately, join the first half of each latent sequence, and decode the result once, over 1,024 OpenWebText pairs. Joining costs the $4\times$ codec 0.45 percentage points of token accuracy, from 99.47% to 99.02%, and costs the two $1\times$ codecs nothing (Figure 5c; intervals and protocol in Appendix E.2). The few extra errors sit where a receptive field crosses a boundary: of 627, 69% lie within 8 tokens of the join, 30% in the last 8 tokens, and fewer than 1% elsewhere (Figure 5b). Latents can therefore be moved between texts, which is exactly what lets the flow hold observed spans fixed. Locality also keeps the latents of observed text independent of the gap: replacing tokens 64–127 of a 128-token text leaves the latents at positions 0–31 of the $1\times$ codec unchanged, so the fixed latents stay consistent with whatever the flow generates in the gap.

4.4 Computation and sampling budget

The cost of a sample depends on how long the latent sequence is and how many times the field processes it. We write the inference time as

$$T_{\text{total}} = T_{\text{encode}} + \sum_{k=1}^{K} T_{\text{field},k} + T_{\text{solver}} + T_{\text{decode}} + T_{\text{other}}. \tag{7}$$

Table 3: Language-modeling perplexity on five held-out benchmarks. Manta-LM uses the 101M OpenWebText flow and reports an ELBO-based upper bound on perplexity. GPT-2-small (80K) is the retrained baseline; GPT-2 is the reference result. Lower is better; column minima are bold. Baselines: D3PM (Austin et al., 2021), PLAID (Gulrajani & Hashimoto, 2023), SEDD (Lou et al., 2024), MD4 (Shi et al., 2024), RADD (Ou et al., 2025), and AO-GPT (Xue et al., 2026).

Model	LAMBADA	WikiText-2	PTB	WikiText-103	1BW
GPT-2	45.04	42.43	138.43	41.60	75.20
GPT-2-small (80K)	54.17	44.07	201.69	43.20	65.50
D3PM	93.47	77.28	200.82	75.16	138.92
PLAID	57.28	51.80	142.60	50.86	91.12
SEDD	50.92	41.84	114.24	40.62	79.29
MD4	48.43	34.94	102.26	35.90	68.10
RADD	51.70	39.98	107.85	37.98	72.99
AO-GPT	54.92	42.24	123.49	41.51	76.73
Manta-LM	**34.80**	**30.58**	**97.76**	**35.35**	**62.55**

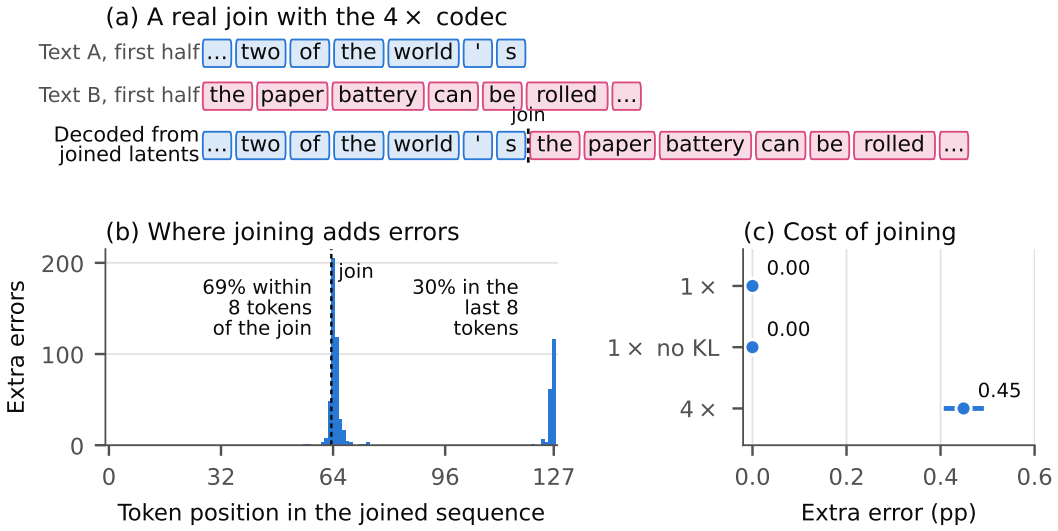


Figure 5: **Latents from different texts can be joined.** (a) A real OpenWebText pair: the first latent halves of texts A and B, encoded separately by the $4\times$ codec, are joined and decoded; this pair's only error is at the last position. (b) Positions of the 627 extra errors over 1,024 pairs: they lie next to the join or at the new end of the sequence, where the receptive field crosses a boundary. (c) Extra error from joining, with paired 95% intervals.

where K counts field evaluations, including those for guidance, and T_{other} covers tokenization and output processing. Compression shortens the sequence in each of the K evaluations, and a smaller step budget reduces K.

Compression makes each step 3.2–$3.5\times$ cheaper. On 128-token sequences, the 101M flow processes 25.2k tokens per second at 30 steps with the $4\times$ codec, against 7.3k with the $1\times$ codec, and the $4\times$ codec lowers the cost of each step by 3.2–$3.5\times$ across 10–100 steps (Figure 6).

The best guidance depends on the step budget. Strong guidance helps long trajectories but fails when only a few updates are available (Figure 7). On Quasar-T, the default $w = 7$ scores BLEU 0.000 at four steps and 0.173 at 64. A schedule selected on 256 validation examples already reaches 0.180 at four steps, above the 64-step default (paired difference $+0.006$, 95% interval $[0.004, 0.008]$) with 4 field evaluations instead of 128, and reaches 0.185 at 64 steps. On QQP, $w = 1$ at four steps gives BLEU 0.182 versus 0.062 for $w = 3$ (Appendix F.1).

The same interaction holds on all four tasks (Figure 8). With $w = 7$, BLEU is competitive at 50 steps but falls sharply at 10 steps, for example from 0.271 to 0.033 on paraphrasing. The best guidance also

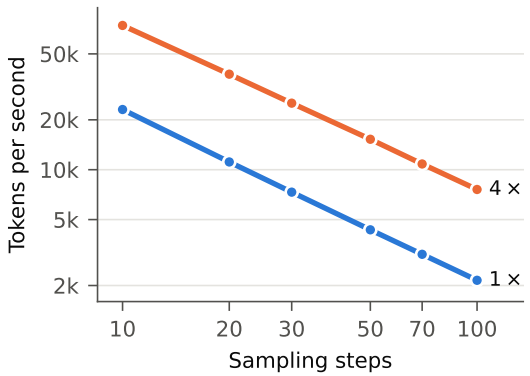


Figure 6: Denoising throughput of the 101M flow on 128-token sequences with the $1\times$ and $4\times$ codecs ($w = 3$, 2,500 sequences per point, one diffusion update per step).

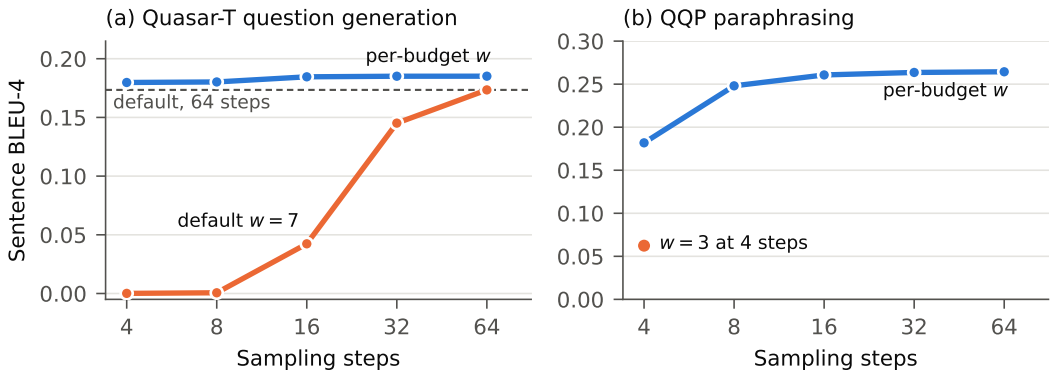


Figure 7: Choosing guidance for each step budget. With guidance selected per budget on validation data, four steps already exceed the default guidance at 64 steps on question generation. (a) Quasar-T, 10,000 test examples: default $w = 7$ versus the selected schedule ($w = 1$ at 4–8 steps, $w = 3$ at 16–64). (b) QQP, 2,500 test examples: the selected schedule ($w = 1$ at 4 steps, $w = 3$ above) and $w = 3$ at 4 steps. Scores are sentence-level BLEU-4.

depends on the task: $w = 5$–7 for paraphrasing and simplification, $w = 3$ for question generation, and $w = 1$ for dialogue. Guidance is therefore a sampling hyperparameter to select per task and per step budget, at no training cost.

4.5 Generation under partial observations

Infilling surpasses larger autoregressive models in MAUVE. Manta-LM generates the missing middle with both the prefix and the suffix observed. On OpenWebText passages of up to 1,024 tokens with the first and last 10% observed, the 0.6B flow reaches MAUVE 0.616, above all four Qwen2.5 models (Qwen et al., 2024), the best of which reaches 0.593 with 3B parameters (Table 4). In the example of Figure 9, the generated middle continues the prefix with the same people, places, and events for several sentences before reaching the observed suffix; Appendix H shows a second passage, with the observed text in blue.

The generated middle follows the suffix. To test whether the middle depends on the suffix, we change only the suffix while keeping the prefix, gap length, and sampling noise fixed. The generated middle changes in 60 of 64 such pairs, across all eight kinds of suffix change we test, such as event order, location, and recipient, and every observed token is preserved (Appendix G).

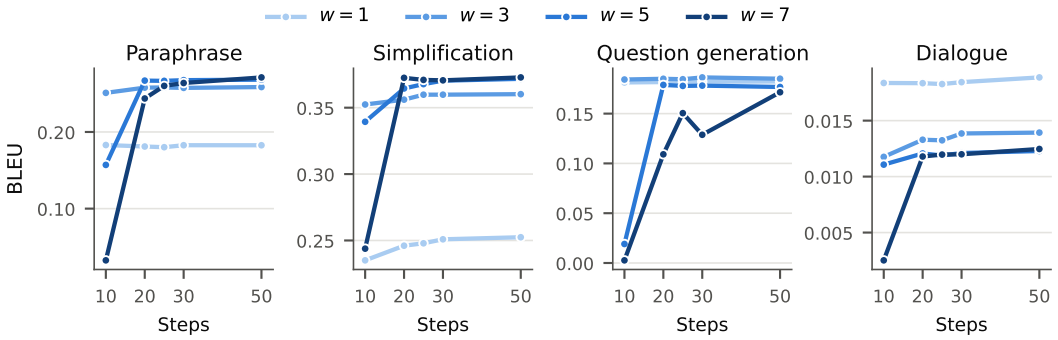


Figure 8: Guidance strength w and number of sampling steps for the four task flows (BLEU). Strong guidance is best or competitive with 20 or more steps but drops sharply at 10 steps on every task.

Rabat – Dutch far-right lawmaker Geert Wilders was found not guilty of hate speech but guilty of discrimination and group insult. He will face no punishment. The verdict is in reference to comments Wilders, the leader of the Freedom Party, made about Moroccans while campaigning in The Hague in 2014.On camera, Wilders said of his campaign: 'They are voting for a safer and more social city with fewer problems and, if possible, fewer Moroccans.' Wilders also asked the audience at a rally if they wanted 'more or less Moroccans' and the crowd responded, as instructed, 'less'. The comments were met with backlash from politicians, Muslim organizations, and Moroccan-Dutch people race and convict me and half of the Netherlands. Madness. — Geert Wilders (@geertwilderspvv) December 9, 2016Wilders questioned the legitimacy of the court, and his lawyer announced that they will appeal the decision.

Figure 9: Prefix–suffix infilling by the 0.6B flow on an OpenWebText passage. Blue text is the observed prefix and suffix; black text is generated.

Table 4: Prefix–suffix infilling on OpenWebText. Higher MAUVE is better.

Model	Size	MAUVE
Qwen2.5	1.5B	.543
Qwen2.5-Inst.	1.5B	.569
Qwen2.5	3B	.593
Qwen2.5-Inst.	3B	.543
Manta-LM	0.6B	**.616**

5 Conclusion

We introduce Manta-LM, a parallel text generator that operates in a compressed latent space while preserving correspondence between latent positions and regions of the text. A convolutional TextVAE compresses text locally, allowing the latents of observed text to be held fixed or transferred across texts. A masked Transformer flow generates the free latents with global attention, and a parallel decoder reads out every token. By changing the observation mask, the same architecture supports unconditional generation, continuation, and infilling. Experimentally, at $4\times$ compression, the codec reconstructs 99.75% of tokens, while joining latents from different texts reduces token accuracy by only 0.45 percentage points, and each flow step runs 3.2–3.5$\times$ faster than at $1\times$. With one 101M-parameter flow per task, Manta-LM obtains the highest BLEU among the compared models on paraphrasing, question generation, simplification, and dialogue. Its OpenWebText flow also achieves the lowest perplexity bound on five zero-shot benchmarks and a Gen PPL of 46.9 with 16 sampling steps, compared to 78.3 for RADD. The 0.6B flow achieves higher MAUVE than Qwen2.5-3B on infilling. Furthermore, selecting guidance jointly with the sampling budget allows four sampling steps to match the performance of 64 steps on question generation. These results suggest that the codec and generator should be designed jointly: preserving local correspondence in the compressed representation turns conditioning on arbitrary text spans into a simple operation of fixing the corresponding latents and generating the rest.

Limitations and future work. Our experiments use flows of up to 0.6B parameters and passages of up to 1,024 tokens. Because compression shortens the sequence over which the attention cost of the flow grows, larger flows on longer documents with many observed spans are a natural next step for this design. Our observation-mask formulation also supports multiple disjoint observed and missing spans within a passage, and it is promising to systematically evaluate such multi-span infilling or more complex document-editing settings. Finally, jointly optimizing the codec and generator, rather than training them as separate components, may further improve the quality and efficiency of the learned latent space in future work.

AI USE STATEMENT

Generative AI tools assisted with manuscript polishing and language editing, literature search, experimental scripting, and processing and visualization of existing outputs. They also assisted in constructing the synthetic templates for the suffix test. The example text in Figure 1 is illustrative. Quantitative plots are generated from recorded measurements. The authors retain responsibility for the final text, references, claims, and experimental results.

ETHICS STATEMENT

Generated text can contain unsupported claims or reproduce biases in the training data. The reported metrics are not guarantees of factual reliability, and applications that fill in or rewrite text should check the generated content.

REPRODUCIBILITY STATEMENT

Section 3 specifies the latent representation, conditional flow, and sampling formulation. The appendix provides architecture and training configurations, evaluation protocols, full result tables, and qualitative examples. The supplementary material contains the numerical results and evaluation protocols.

REFERENCES

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

Jacob Austin, Daniel D Johnson, Jonathan Ho, Marc Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*, 2022.

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. In *Advances in neural information processing systems*, volume 33, pp. 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. In *Interspeech*, 2013.

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *International Conference on Learning Representations*, 2025.

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.

Chaoran Cheng, Jiahan Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical manifolds. *Advances in Neural Information Processing Systems*, 37:54787–54819, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DataCanary, hilfialkaff, Lili Jiang, Meg Risdal, Nikhil Dandekar, and tomtung. Quora question pairs. Kaggle Competition, 2017. https://kaggle.com/competitions/quora-question-pairs.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*, 2017.

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.

Chris Donahue, Mina Lee, and Percy Liang. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2492–2501, 2020.

Wanyu Du, Jianqiao Zhao, Liwei Wang, and Yangfeng Ji. Diverse text generation via variational encoder-decoder models with Gaussian process priors. In *Proceedings of the Sixth Workshop on Structured Prediction for NLP*, 2022.

Zhicheng Du and Lan Ma. Continuous language diffusion as a decoder-interface problem. *arXiv preprint arXiv:2606.08810*, 2026.

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*, 2023.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. *Advances in neural information processing systems*, 32, 2019.

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

Hongcan Guo, Qinyu Zhao, Yian Zhao, Shen Nie, Rui Zhu, Qiushan Guo, Feng Wang, Tao Yang, Hengshuang Zhao, Guoqiang Wei, and Yan Zeng. Continuous latent diffusion language model. *arXiv preprint arXiv:2605.06548*, 2026.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

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.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Weidi Xu, Jiaran Hao, Liuyihan Song, Yang Xu, Jian Yang, Jiaheng Liu, Chenchen Zhang, et al. Opencoder: The open cookbook for top-tier code large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33167–33193, 2025.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. Neural CRF model for sentence alignment in text simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7943–7960, 2020.

Jiaxiu Jiang, Jingjing Ren, Wenbo Li, Bo Wang, Haoze Sun, Yijun Yang, Jianhui Liu, Yanbing Zhang, Shenghe Zheng, Yuan Zhang, Haoyang Huang, Nan Duan, and Wangmeng Zuo. TextLDM: Language modeling with continuous latent diffusion. *arXiv preprint arXiv:2605.07748*, 2026.

Jaehyeong Jo and Sung Ju Hwang. Continuous diffusion model for language modeling. In *Neural Information Processing Systems*, 2025.

Jijie Li, Li Du, Hanyu Zhao, Bo-wen Zhang, Liangdong Wang, Boyan Gao, Guang Liu, and Yonghua Lin. Infinity instruct: Scaling instruction selection and synthesis to enhance language models. *arXiv preprint arXiv:2506.11116*, 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.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate with rectified flow. In *International Conference on Learning Representations*, 2023.

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

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry Vetrov. COSMOS: Compressed and smooth latent space for text diffusion modeling. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

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 *Neural Information Processing Systems*, 2025.

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.

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.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Krishna Pillutla, Lang Liu, John Thickstun, Sean Welleck, Swabha Swayamdipta, Rowan Zellers, Sewoong Oh, Yejin Choi, and Zaid Harchaoui. Mauve scores for generative models: Theory and practice. *Journal of Machine Learning Research*, 24(356):1–92, 2023.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

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

Chenze Shao, Darren Li, Fandong Meng, and Jie Zhou. Continuous autoregressive language models. *arXiv preprint arXiv:2510.27688*, 2025.

Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi Jaakkola. Blank language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 5186–5198, 2020.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167, 2024.

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.

Shuchen Xue, Tianyu Xie, Tianyang Hu, Zijin Feng, Jiacheng Sun, Kenji Kawaguchi, Zhenguo Li, and Zhi-Ming Ma. Any-order GPT as masked diffusion model: Decoupling formulation and architecture. In *International Conference on Machine Learning*, 2026.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. Text diffusion model with encoder-decoder transformers for sequence-to-sequence generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 22–39, 2024.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*, 2020.

Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Joshua Susskind, and Navdeep Jaitly. PLANNER: Generating diversified paragraph via latent language diffusion model. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, volume 18, pp. 4623–4629, 2018.

A Implementation and training

A.1 TextVAE

TextVAE adapts the design of a deep compression autoencoder (Chen et al., 2025) from images to one-dimensional token sequences. The encoder predicts the mean and diagonal variance in Equation 1. The convolutional decoder uses residual blocks to map sampled latents to token embeddings of shape $B \times L \times d_h$, where B, L, and d_h denote batch size, output length, and embedding width. Dot products with the shared vocabulary embedding matrix, followed by a learned scale and bias, produce vocabulary logits. Both networks are parallel and omit self-attention. Two downsampling stages provide $4\times$ sequence compression. Finite, overlapping receptive fields associate each latent region with a bounded text neighborhood; adjacent groups can share information through that overlap.

The reconstruction term has coefficient 1, the final KL weight is 10^{-2}, and the feature-stability coefficient is 10^{-4}. Let a_{biv} be the decoder logit for vocabulary item v at position i of sample b, and y_{biv} its one-hot target. The implementation uses

$$a_{biv} = e^s \langle d_{bi}, W_v \rangle + b_0, \qquad \mathcal{L}_{\text{rec}} = \frac{1}{B} \sum_b \frac{\sum_i w_{bi} \sum_v \text{BCEWithLogits}(a_{biv}, y_{biv})}{\sum_i w_{bi}}, \tag{8}$$

where $w_{bi} = 10$ for non-PAD tokens and 1 for PAD tokens. The vocabulary dimension is summed, the token weights are normalized within each sample, and the batch is averaged. At evaluation, a softmax over the logits defines the categorical decoder distribution.

For captured intermediate activations $f^{(1)}, \ldots, f^{(J)}$, the stability term penalizes unusually large magnitudes:

$$\mathcal{L}_{\text{stab}} = \frac{1}{J} \sum_{j=1}^{J} \frac{1}{|f^{(j)}|} \sum_k \max\big(|f_k^{(j)}| - 100, 0\big). \tag{9}$$

The KL is summed over latent positions and channels and averaged over samples. Its weight starts at 10^{-4} and increases linearly to $\beta_{\max}$ over one third of the N training steps, so posterior regularization is introduced gradually while reconstruction is learned.

Table 5: Model architectures. Codec widths and block counts follow encoder order; decoder stages reverse it. Compression denotes sequence-length reduction; flow parameter counts exclude the codec.

Component	Configuration
BERT $1\times$ codec	16 latent channels; stage widths 512, 1024; three residual blocks per stage; no sequence downsampling
BERT $4\times$ codec	16 latent channels; widths 512, 512, 1024, 1024, 1024, 1024; two residual blocks per stage; two factor-two downsampling stages
GPT-2 $1\times$ codec	16 latent channels; widths 512, 1024; three residual blocks per stage; no sequence downsampling
101M flow	12 layers; width 768; FFN width 3072; 6 query / 3 KV heads; head width 128; 101,184,016 parameters
0.6B-family flow	24 layers; width 1280; FFN width 4992; 14 query / 2 KV heads; head width 128; 556,093,968 parameters

Table 6: Parameter counts of complete pipelines. Counts include the shared vocabulary embedding once. The complete encode–flow–decode path uses both components; each refinement evaluation uses only the flow. The 101M and 0.6B labels denote flow size.

Pipeline	Flow	Codec	Full pipeline
101M task flow; BERT $1\times$ codec	101,184,016	72,583,458	173,767,474
0.6B flow; GPT-2 $1\times$ codec	556,093,968	87,721,506	643,815,474

The codec is trained separately and frozen during flow training. Table 5 gives the flow architectures.

A.2 Data and model training

The codec training corpus includes general text, code, and mathematics: Wiki, Common Crawl and LM1B; OpenCoder and Infinity-Instruct; and OpenMathReasoning and GSM8K (Chelba et al., 2013; Huang et al., 2025; Li et al., 2025; Moshkov et al., 2025; Cobbe et al., 2021). The codec is trained first and then frozen; the flows are trained afterwards on their own data.

Each conditional task (paraphrasing, question generation, simplification, dialogue) has its own 101M flow trained on that task's training pairs and evaluated on its held-out test split without further weight updates. Language modeling uses the OpenWebText processing and benchmark conventions of established protocols (Gokaslan & Cohen, 2019; Lou et al., 2024; Ou et al., 2025). The 0.6B flow is trained on SlimPajama and used for infilling and the suffix test.

Table 7: Training stages and the results each model produces.

Stage / group	Data and objective	Evaluation usage
Codec pretraining	Text, code, mathematics; reconstruction, KL, activation penalty	Local representation and parallel decoder
Task flows (101M)	One flow per conditional task, trained on its paired data; BERT $1\times$ codec; conditional flow matching	Table 2; guidance study (Figures 7 and 8)
Unconditional flow (101M)	OpenWebText; GPT-2 $1\times$ codec	Unconditional generation (Figure 3), language modeling (Table 3)
0.6B flow	SlimPajama; GPT-2 $1\times$ codec	Infilling (Table 4), suffix test
GPT-2-small baseline	Same tokenizer, data, and schedule as the corresponding Manta comparison	Autoregressive baseline trained for 80,000 updates

Retrained autoregressive baseline. The GPT-2-small (80K) row in Table 3 uses the same tokenizer, training data, learning-rate schedule, and other training hyperparameters as the corresponding Manta comparison, with 80,000 updates.

A.3 Flow training details

Latents are standardized with a scalar mean and standard deviation specific to each model.

Time sampling and regression. Training times are drawn from a logit-normal distribution, $r = \mathrm{sigmoid}(g)$ with $g \sim \mathcal{N}(0, 1)$, followed by a resolution-dependent shift

$$t = \frac{e^{\mu(q)}}{e^{\mu(q)} + 1/r - 1}, \qquad \mu(q) = 0.5 + \frac{q - 256}{4096 - 256}(1.15 - 0.5), \tag{10}$$

where q is the resolution supplied to the scheduler. Regression targets the clean-to-noise velocity. The squared error is computed on free positions only, normalized per sample, with padding positions down-weighted.

A.4 Conditioning and inference

Table 8 distinguishes information available at inference. The mask selects separate linear projections for context and free latents. Adaptive normalization supplies time information; bidirectional attention mixes free states and observed context.

Groupwise boundary alignment. The main $4\times$ configuration pads observed and free token spans to compression-group boundaries. For example, the mask `C__CCC____` is packed as `|C000|__00|CCC0|____|`, where `C` marks an observation, `_` a free slot, and `0` a PAD token. Observed groups provide context latents; free groups are initialized from noise for generation. The decoder removes padding when assembling the output text. The composition test in Appendix E evaluates joining latent segments from independently encoded texts.

Packing. Conditional tasks tokenize source and target regions separately, assemble a token canvas, and encode it before selecting observed and free groups. Overlength pairs are shortened from the longer segment until they fit. The source-length boundary defines the latent mask.

Infilling uses a prefix–gap–suffix canvas: the gap is filled with padding, the canvas is encoded, and the free latent groups are replaced with noise. Gap indices map directly to latent indices.

Output assembly. With $m_i = 1$ at free latent positions, each field input is assembled as $m \odot z_t + (1 - m) \odot h_C$. The final sequence is assembled in the same way, un-standardized, and decoded with a per-position argmax over the vocabulary; the task region is then extracted and special tokens are removed. Observed tokens are decoded from their latents like all other positions.

Table 8: Generation modes. A shared model changes its conditioning and initial state according to the observation pattern.

Mode	Observed input	Initial free state
Generation	No text observations	Gaussian noise
Continuation	Prefix	Gaussian noise in the remaining positions
Infilling	Prefix and suffix	Gaussian noise in the missing region

For generation from noise, the clean-at-zero convention integrates from $t = 1$ to $t = 0$. With observed features h_C fixed, the common sampling operation is

$$\widehat{z}_{0,F} = \mathrm{ODESolve}(v_w(\cdot, h_C, t), z_{t_{\mathrm{start}},F}, t_{\mathrm{start}}, 0)\,, \tag{11}$$

followed by parallel decoding with the context states. Infilling uses 70 Euler updates with $w = 1$. Guidance with $w \neq 1$ doubles the field evaluations per step.

B Comparison of design choices

Table 9 compares the representation, update structure, and conditioning studied by related continuous language models. Sequence compression and local addressability are different properties: a global compressor can aggregate an entire passage into each latent, whereas a local codec associates latent groups with bounded token neighborhoods. Manta uses the latter organization and a global flow to coordinate the generated regions.

Our experiments evaluate the two parts together: reconstruction and joining for the local codec, and conditional generation, unconditional generation, infilling, and sampling budgets for the global flow. Related analyses study the decoder interface (Du & Ma, 2026); rectified flow provides a related straight-path generative construction (Liu et al., 2023).

C Likelihood and evaluation conventions

Density estimation. The latent-variable model combines a categorical decoder likelihood, a continuous flow-prior density, and a Gaussian encoder posterior:

$$\log p(x) \geq \mathcal{L}_{\mathrm{ELBO}}(x) = \mathbb{E}_{q_\phi(z_0|x)}[\log p_\psi(x \mid z_0) + \log p_\theta(z_0) - \log q_\phi(z_0 \mid x)]. \tag{12}$$

For decoder logits a_{iv} and token x_i, the likelihood is

$$\log p_\psi(x \mid z_0) = \sum_{i=1}^{L} \left(a_{i,x_i} - \log \sum_{v \in \mathcal{V}} \exp a_{iv} \right). \tag{13}$$

The Gaussian posterior log density sums over latent positions and channels. For the clean-at-zero flow convention, change of variables gives

$$\log p_\theta(z_0) = \log \mathcal{N}(z_1; 0, I) + \int_0^1 \mathrm{tr}\left(\frac{\partial v_\theta(z_t, t)}{\partial z_t} \right) dt. \tag{14}$$

The prior and posterior densities must be expressed in the same coordinates: an affine latent normalization contributes the corresponding log-Jacobian to both densities, which cancels in their difference. All logarithms are natural. Negating the ELBO and dividing by the scored token count L gives nats

Table 9: Structural comparison with related methods. Conditioning entries describe the formulations and applications in the cited works, rather than ruling out other extensions. "Parallel" refers to simultaneous position-wise output within the specified sequence or chunk.

Method	Codec and regional organization	Generator / decoding	Conditioning studied
COSMOS (2025)	Contextual encoder and global Perceiver compression	Latent diffusion; parallel decompression and token prediction	Conditional generative tasks, including question generation and summarization
TextLDM (2026)	Transformer VAE; one latent per token; context and target encoded separately	Whole-target flow; parallel Transformer decoder	Clean prefix and noisy target; conditional and unconditional generation
Cola-DLM (2026)	Causal TextVAE; token-aligned, uncompressed sequence in experiments	Block-causal flow; causal conditional decoder	Prefix-conditioned, blockwise generation
CALM (2025)	Context-free chunk autoencoder; explicit chunk addresses	Autoregressive latent generation; parallel decoding within each chunk	Previous chunks condition the next chunk
ELF (2026)	Token-aligned embeddings, contextual T5 by default	Whole-target flow; shared-weight final decoding	Clean prefix embeddings; translation and summarization
LangFlow (2026)	Learned token embeddings; no sequence compression	Embedding-space flow with self-conditioning; parallel token prediction	Unconditional language modeling in the reported benchmark recipe
Manta-LM	Local convolutional codec; $1\times$ / $4\times$ states with regional correspondence	Global conditional flow; fully parallel codec decoder	Observed/free masks; generation, continuation, prefix–suffix infilling

per token; exponentiation gives the perplexity bound, regardless of latent compression. Numerical posterior sampling and ODE integration approximate this expression. Gen PPL instead scores generated text with GPT-2 Large.

Task metrics. Conditional generation uses BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004), BERTScore (Zhang et al., 2020), and Distinct-1. Infilling uses MAUVE (Pillutla et al., 2023).

Continuous-baseline density results. Table 10 gives additional density estimates for DiffuSeq and SeqDiffuSeq. Both models were designed for conditional generation.

Table 10: Additional continuous-baseline perplexity estimates. Lower is better.

Model	LAMBADA	WikiText-2	PTB	WikiText-103	1BW
DiffuSeq	148.63	115.34	378.42	122.87	164.29
SeqDiffuSeq	112.38	88.75	291.56	94.10	129.85

D Full generation results

D.1 Unconditional generation

D.2 Conditional generation

Tables 12–15 give the four quality and diversity metrics by task. FT denotes a fine-tuned baseline; each Manta-LM column uses the flow trained for that task (Section 3.3). Parameter counts are listed in Table 12.

Table 11: The complete reported unconditional generation sweep. Gen PPL, lower is better; unigram entropy measures lexical diversity.

Model	Metric	16	32	64	128	256
MDLM	Gen PPL	137.02	120.14	103.37	82.64	66.28
RADD	Gen PPL	78.29	60.49	46.23	37.53	32.68
Manta-LM	Gen PPL	46.92	32.10	29.18	23.80	23.56
RADD	Entropy	6.1076	6.3109	6.3501	6.4133	6.4214
Manta-LM	Entropy	6.1232	6.0459	5.7981	6.0382	5.9714

Table 12: Paraphrase generation. All four metrics are higher-is-better. A dash indicates an unreported entry.

Model	Parameters	BLEU	ROUGE-L	BERTScore	Distinct-1
GPT-2 base FT	117M	.198	.521	.825	.980
GPT-2 large FT	774M	.206	.542	.836	.982
GPVAE-T5	220M	.241	.589	.847	.969
LevT	80M	.227	.580	.834	.979
DiffuSeq	91M	.241	.588	.837	.981
SeqDiffuSeq	108M	.243	–	.840	.981
RDLM	–	.215	.482	.704	–
Manta-LM	101M	.301	.653	.848	.971

Table 13: Question generation.

Model	BLEU	ROUGE-L	BERTScore	Distinct-1
GPT-2 base FT	.074	.271	.605	.960
GPT-2 large FT	.111	.322	.635	.967
GPVAE-T5	.125	.339	.631	.938
LevT	.093	.289	.549	.891
DiffuSeq	.173	.367	.612	.906
SeqDiffuSeq	.175	–	.617	.925
RDLM	.179	.315	.576	–
Manta-LM	.186	.378	.613	.880

Table 14: Text simplification.

Model	BLEU	ROUGE-L	BERTScore	Distinct-1
GPT-2 base FT	.308	.546	.802	.944
GPT-2 large FT	.269	.511	.788	.946
GPVAE-T5	.339	.583	.817	.931
LevT	.205	.440	.725	.972
DiffuSeq	.362	.585	.813	.926
SeqDiffuSeq	.371	–	.821	.908
RDLM	.280	.467	.687	–
Manta-LM	.389	.613	.822	.904

Table 15: Open-domain dialogue.

Model	BLEU	ROUGE-L	BERTScore	Distinct-1
GPT-2 base FT	.011	.151	.528	.919
GPT-2 large FT	.013	.100	.529	.924
GPVAE-T5	.011	.101	.432	.563
LevT	.016	.055	.476	.973
DiffuSeq	.014	.106	.513	.947
SeqDiffuSeq	.011	–	.443	.961
RDLM	.011	.109	.481	–
Manta-LM	.020	.112	.485	.933

E REPRESENTATION EVALUATION

E.1 TOKEN RECONSTRUCTION

Table 16 evaluates token reconstruction across 19 datasets covering general text, code, and mathematics. At $4\times$ compression, all listed accuracies exceed 99.3%, with an average of 99.75%. The $1\times$ codec achieves 1.0000 at the displayed precision on every dataset.

Table 16: Token reconstruction accuracy of the $1\times$ and $4\times$ codecs on 19 datasets.

Dataset	$1\times$	$4\times$	Dataset	$1\times$	$4\times$
HumanEval	1.0000	.9954	CEval	1.0000	.9977
MBPP	1.0000	.9980	MMLU	1.0000	.9935
BBH	1.0000	.9973	GSM8K	1.0000	.9968
ARC	1.0000	.9989	Math	1.0000	.9967
TruthfulQA	1.0000	.9973	LM1B	1.0000	.9997
Winogrande	1.0000	.9996	CodeAlpaca	1.0000	.9990
HellaSwag	1.0000	.9960	QQP	1.0000	.9977
PIQA	1.0000	.9988	Wiki	1.0000	.9962
CMMLU	1.0000	.9983	Common Crawl	1.0000	.9973
			Quasar-T	1.0000	.9985

E.2 PAIRED ORDINARY AND SPLICED RECONSTRUCTION

The composition test directly evaluates the design goal in Section 3.1: corresponding latent spans should remain usable when joined. We compare ordinary and spliced reconstruction on identical retained source positions, separating the extra cost of composition from each codec's reconstruction error. Three frozen codecs share a set of 1,024 text pairs, separate from the 19-dataset reconstruction evaluation.

Inputs and composition. We select 2,048 OpenWebText documents with a fixed random seed, exclude duplicate documents and duplicate token blocks, and retain the first 128 BERT wordpieces without adding special tokens. Consecutive documents form pairs (A, B). Each codec independently encodes both full texts using the posterior mean. For ordinary reconstruction, we decode both sequences and retain their first 64 positions. For spliced reconstruction, we concatenate their first latent halves and decode once, targeting $A_{1:64} \| B_{1:64}$. The join is aligned to a latent-group boundary at both compression factors.

Scoring. Both conditions score exactly the same reference positions, excluding all tokenizer special tokens, including PAD and UNK. Accuracy pools correct tokens over the 131,070 eligible positions. Extra loss is ordinary minus spliced accuracy. We resample text pairs 10,000 times with replacement, preserving both conditions within each pair and using the pooled token denominator in each resample. Table 17 reports percentile 95% intervals.

Table 17: Paired composition on 1,024 OpenWebText pairs. Each cell gives an estimate and its 95% interval. Accuracy is in percent; extra loss is in percentage points (pp). The same source positions are scored before and after composition.

Codec	Compression	Ordinary (%)	Spliced (%)	Extra loss (pp)
Local, with KL	$1\times$	99.997 [99.994, 99.999]	99.997 [99.994, 99.999]	0.000 [0.000, 0.000]
Local, without KL	$1\times$	99.999 [99.998, 100.000]	99.999 [99.998, 100.000]	0.000 [0.000, 0.000]
Local, compressed	$4\times$	99.471 [99.415, 99.526]	99.021 [98.955, 99.087]	0.449 [0.412, 0.487]

The $4\times$ codec keeps 99.02% of tokens after joining, 0.45 percentage points below ordinary reconstruction, and both $1\times$ codecs keep their ordinary accuracy in every pair.

Codec configuration. All three codecs use the BERT-base-uncased tokenizer and 16 latent channels. The $1\times$ codecs have 72.6M parameters each; the $4\times$ codec has 154.2M parameters and maps 128 tokens to 32 latent positions.

F GUIDANCE AND SAMPLING BUDGET

F.1 BUDGET STUDY

This study uses the 101M task flows without additional training. Test budgets are $\{4, 8, 16, 32, 64\}$ steps. BLEU is the mean NLTK sentence-level BLEU-4 with smoothing method 4, and ROUGE-L is the `rouge_score` F1, against the original references.

For Quasar-T, we evaluate $w \in \{1, 3, 7\}$ at five budgets on 256 validation examples and, at each budget, select the w with the highest BLEU, breaking ties by ROUGE-L and then smaller w. The resulting schedule, $w = 1$ at 4 and 8 steps and $w = 3$ at 16, 32, and 64 steps, is fixed before evaluation on all 10,000 test inputs. Paired 95% intervals use 2,000 resamples of the test inputs. Four steps with $w = 1$ need 4 field evaluations, compared with 128 for 64 steps with $w = 7$.

QQP uses 256 validation examples and $w \in \{1, 3\}$; the same rule selects $w = 1$ at four steps and $w = 3$ above, evaluated on all 2,500 test inputs. At four steps, $w = 1$ improves BLEU from 0.062 to 0.182 over $w = 3$ (paired 95% interval of the difference $[0.114, 0.125]$). Dialogue keeps $w = 1$ on all 10,000 test inputs and changes little with the budget (Table 18).

Table 18: Budget sweeps of the task flows. BLEU is mean sentence-level BLEU-4 (smoothing method 4); RL is ROUGE-L F1. Quasar-T (QT) and dialogue use 10,000 test examples each and QQP uses 2,500. The selected guidance is $w = 1, 1, 3, 3, 3$ for QT and $w = 1, 3, 3, 3, 3$ for QQP.

Task	Guidance	Metric	4	8	16	32	64
QT	$w = 7$	BLEU	0.000	0.001	0.042	0.145	0.173
QT	$w = 7$	RL	0.001	0.003	0.135	0.318	0.360
QT	Selected	BLEU	0.180	0.180	0.185	0.185	0.185
QT	Selected	RL	0.356	0.357	0.369	0.369	0.369
QQP	$w = 3$	BLEU	0.062	0.248	0.261	0.263	0.264
QQP	$w = 3$	RL	0.241	0.606	0.623	0.624	0.625
QQP	Selected	BLEU	0.182	0.248	0.261	0.264	0.264
QQP	Selected	RL	0.515	0.606	0.622	0.624	0.625
Dialogue	$w = 1$	BLEU	0.017	0.018	0.018	0.019	0.019
Dialogue	$w = 1$	RL	0.094	0.096	0.097	0.097	0.098

F.2 COMPUTATIONAL COST

Table 19 reports training cost and per-example inference time of a 101M task flow on a single NVIDIA RTX A6000, with batch size 64 and sequence length 128.

Table 19: Training and inference cost summary. Inference uses batch size 64 and sequence length 128.

Setting or measurement	Value
Training updates	80,000
Training devices	$2\times$ RTX A6000
Training cost	658.6 GPU hours
Inference memory	4.2 GB
Time per example	29.33 ms
Time per step	1.47 ms

G SUFFIX TEST

We construct 64 pairs of inputs in eight families: assignment, decision, direction, event order, location, recipient, schedule, and state, with eight pairs per family. The two inputs of a pair share the prefix and gap length but differ in the suffix. Both are generated by the 0.6B flow from the same Gaussian noise, with 70 Euler steps and $w = 1$.

The generated gap changes in 60 of 64 pairs, and the observed tokens are reproduced in all 128 outputs. Changed pairs per family are 8/8/7/8/8/7/6/8 in the order above.

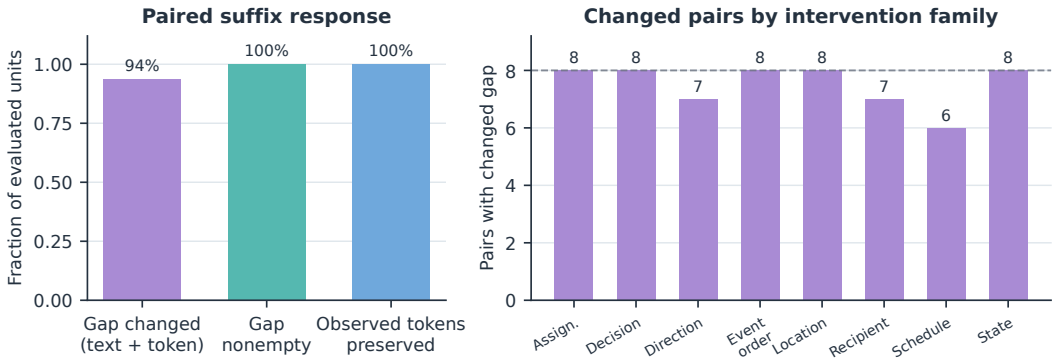


Figure 10: Suffix test. Left: share of pairs whose gap changes with the suffix, and of outputs with a nonempty gap and preserved observed tokens. Right: changed pairs per family.

H QUALITATIVE EXAMPLES

The following examples show intermediate decodings during conditional generation and passages generated by infilling.

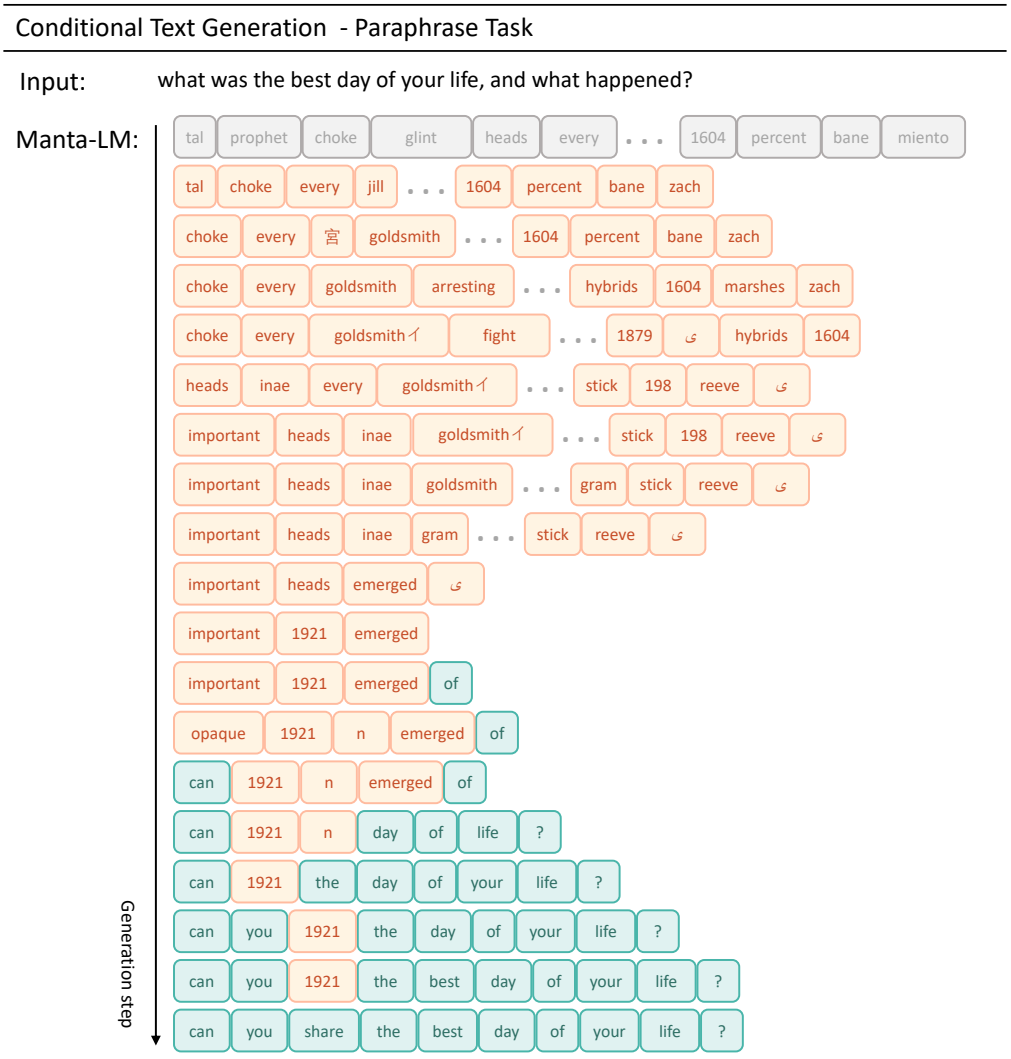


Figure 11: Intermediate paraphrase decodings for the input “what was the best day of your life, and what happened?” Positions are revised jointly across steps.

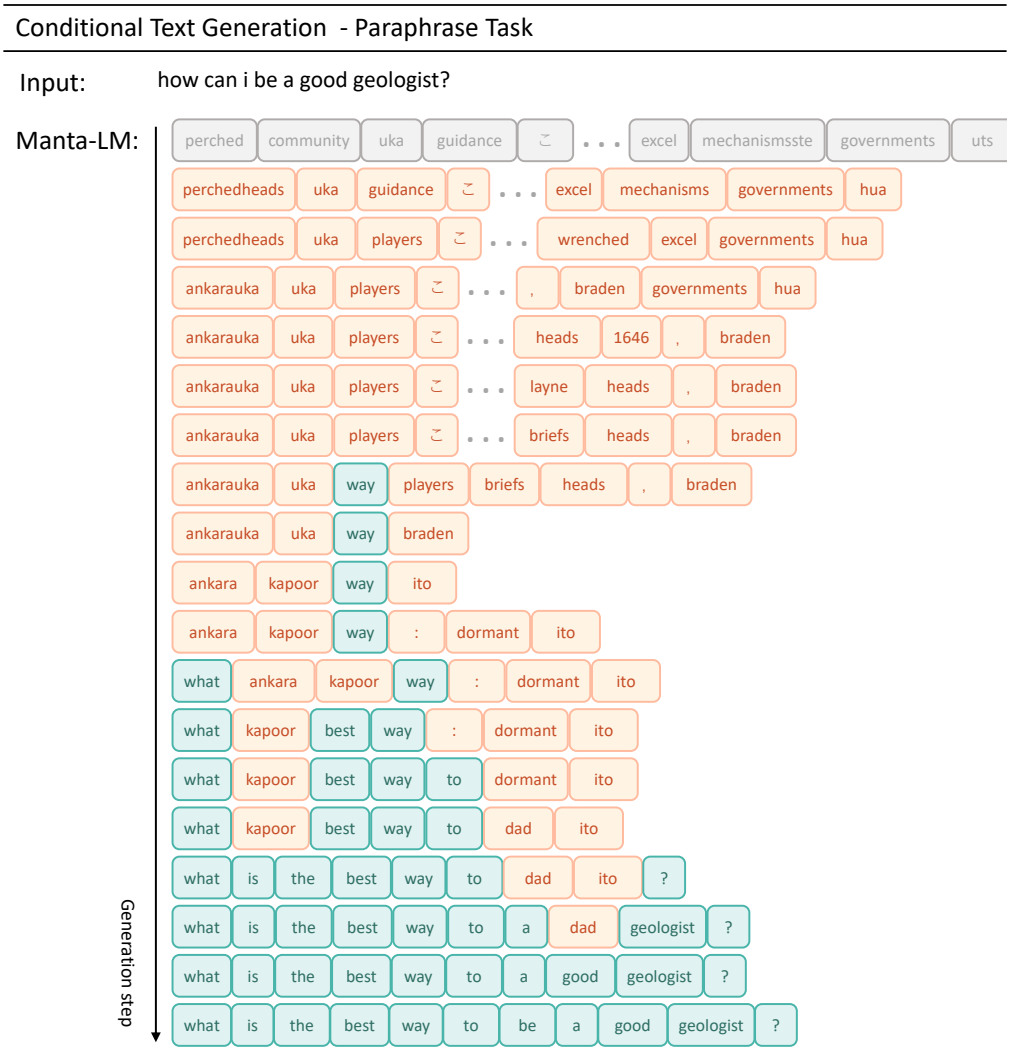


Figure 12: Intermediate paraphrase decodings for “how can i be a good geologist?” Multiple positions are revised before the final decoding.

Wests Tigers five-eighth Mitchell Moses has let the fans down after jumping ship to Parramatta mid-season. (AAP)
May 17th, 2017
However, there is light at the end of the tunnel.
I am looking at the recruitment announcements over recent weeks since new coach Ivan Cleary was appointed at the club. I can see a new team building already.
Before I get to that though, just a quick comment on the transfer of 5/8 Mitchell Moses to the Parramatta Eels this week.
This is a disappointing result for all concerned. I don't buy into the notion that Mitchell should be so distracted by his contract situation that he could not perform for his current employers. There have been plenty of examples over the years where players have signed with rival clubs, sometimes 12 months or more before their current contract expires, yet they have been able to continue playing and giving 100% effort to their current club. endure in recent times, I have no doubt there will be better days ahead.
One also hopes that the Wests Tigers club has learned a real lesson from the events of the last 18 months. Their aim should be that the club should never have to go through this pain again.
In the meantime, let's hope the boys rally together and put some good football on the field for the remainder of the 2017 season. The fans deserve it. They need to cheer for someone.

Figure 13: Infilling example. Blue text is observed context; black text is generated.