One-step Nonautoregressive Natural Language Generation with Shortcut Flow Matching Models

Jędrzej Warczyński[1] **and Ondřej Dušek**[2] **and Mateusz Lango**[1,2]
[1]Poznan University of Technology, Faculty of Computing and Telecommunications, Poznan, Poland
[2]Charles University, Faculty of Mathematics and Physics, Prague, Czechia
jedrzej.warczynski@student.put.edu.pl, {odusek,lango}@ufal.mff.cuni.cz

Abstract

While having a significant potential for parallel processing in theory, diffusion-based nonautoregressive text generation remains inefficient due to the need for multiple denoising steps. Performance degrades sharply if a low number of steps is used, such as in flow matching. To enable accurate one-step generation, we propose a novel *shortcut flow-matching model* that learns to directly predict multi-step denoising outcomes in a single step. Experiments conducted on three datasets demonstrate consistent improvements over classic flow-matching, with BLEU scores more than doubling on two datasets. We also tested five different ways of extending shortcut models with commonly used techniques.

1 Introduction

Non-autoregressive (NAR) text generation has emerged as a promising direction for accelerating language models via fully parallel token processing. Recent diffusion-based approaches have narrowed the gap between the output text quality of NAR and classic autoregressive models (Tae et al., 2025; Zhu et al., 2025).

Despite some advances, diffusion-style models still require numerous iterative denoising steps to numerically solve the underlying stochastic differential equation, which defines a continuous trajectory in latent space along which noise is gradually transformed into data. A high number of steps often eliminates the computational gain from predicting all tokens in parallel (Li et al., 2023; Xu et al., 2025), while reducing the number of steps introduces time-discretization errors: coarse solvers poorly approximate the continuous trajectory, a problem that is further exacerbated by the highly curved flows these models tend to learn in embedding space (Lee et al., 2023).

Recently, flow-matching models (Lipman et al., 2023) emerged as a promising approach for few-step or even one-step text generation. Unlike diffusion models, these methods follow deterministic trajectories instead of relying on stochastic sampling, which makes the denoising process both conceptually simpler and computationally more efficient. Nevertheless, the induced denoising trajectories are still curved and their one-step generation capabilities remains limited, typically falling short of that achieved by diffusion models (Liu et al., 2024; Hu et al., 2024a).

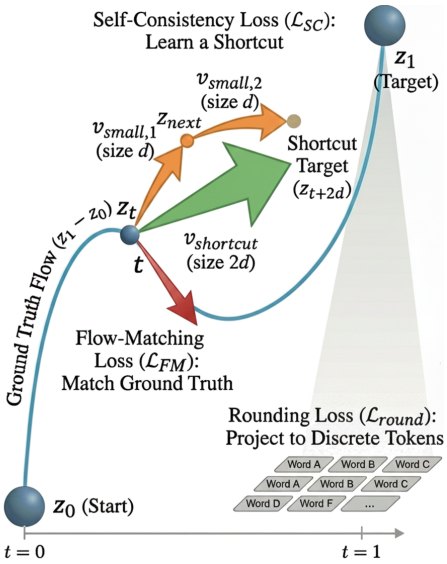

Figure 1: Overview of the proposed method, with visualizations of different components of the loss function.

In this work, we apply the idea of learning shortcuts in flow-matching trajectory (Frans et al., 2025) to enable more accurate NAR text generation with a single denoising step. During training, the proposed shortcut model learns to predict the outcome of multiple denoising steps in a single shot, in addition to performing a standard flow-matching update (see Figure 1). This allows the model to exploit shortcut directions and complete the generation process in far fewer steps, ultimately supporting efficient one-step, non-autoregressive text generation.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 646–655
July 2-7, 2026

Extensive experiments[1] conducted on three text generation datasets demonstrate consistent and significant improvements in text quality over standard flow-matching models.

2 Proposed approach

2.1 Flow-matching model

Flow-matching models generate text starting from a random text representation following Gaussian distribution and iteratively denoise it to match the text representation. Unlike standard diffusion models, whose denoising trajectories inject random noise at every step, flow-matching uses a deterministic linear interpolation to define a path between the random input and the target output. More precisely, the trajectory is

$$z_t = (1-t)z_0 + tz_1 \tag{1}$$

where $z_0 \sim N(0, I)$, $z_1 \sim p_{data}(z)$ and $t \in [0, 1]$. The model learns to approximate the corresponding velocity field,

$$f_\theta(z_t, t) \approx v(z_t, t) = z_1 - z_0 \tag{2}$$

Generation proceeds by sampling a random $z_0 \sim N(0, I)$ and performing n iterative updates:

$$z_{t+\frac{1}{n}} = z_t + \frac{1}{n} f_\theta(z_t, t) \tag{3}$$

In the sequence-to-sequence text generation setting, we denote the source text as x and the target text as y_1. Following the parametrization of Liu et al. (2024), we define the transport flow only over the target sequence – note that x is not dependent on t:

$$z_t = x \oplus y_t$$

where $\oplus$ denotes concatenation.

2.2 Shortcut model

Although standard flow-matching models can produce high-quality text, they often require more denoising steps to match the performance of diffusion models (Liu et al., 2024). To enable more accurate one-step text generation within the flow-matching framework, we propose leveraging shortcut models – originally developed for computer vision (Frans et al., 2025) – to learn to generalize across a broad range of step sizes, including very large ones. Such models were also already successfully applied in the speech domain (Zuo et al., 2025) and the adaptation of these models to text domain is a main contribution of this work.

The key idea of a shortcut model is that, in addition to modeling the standard flow-matching trajectory, the model is also trained to predict shortcut directions, i.e. directions that account for the curvature of the denoising trajectory and allow taking larger steps while remaining on the intended path. The shortcut model is trained by enforcing self-consistency: a single large step taken along a shortcut direction should produce the same result as two smaller steps taken along standard directions.

In comparison to classical flow-matching, the shortcut model $s_\theta(z_t, t, d)$ has an additional parameter d controlling the step size of the shortcut. When $d = 0$, the model reduces to a standard flow-matching model that approximates the instantaneous velocity:

$$s_\theta(z_t, t, 0) \approx v(z_t, t) = z_1 - z_0 \tag{4}$$

For larger step sizes $d > 0$, however, the model must account for the future curvature of the trajectory and predict an appropriate direction for a longer update. In particular, the velocity direction for a step size $2d$ should be equal to the velocity obtained by performing two steps of size d.

$$s_\theta(z_t, t, 2d) = \frac{1}{2}\left(s_\theta(z_t, t, d) + s_\theta(z_{t+d}, t+d, d)\right) \tag{5}$$

2.3 Model training

The above property is used during training in the loss function that combines three terms: (1) **Flow-Matching Loss** ($\mathcal{L}_{\text{FM}}$) for local steps ($d \approx 0$), (2) **Self-Consistency Loss** ($\mathcal{L}_{\text{SC}}$) for recursive bootstrapping across larger d, and (3) **Rounding Loss** ($\mathcal{L}_{\text{round}}$) as a regularizer to prevent embedding collapse. The full training loss is a weighted sum of the terms described above:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{FM}} + \lambda_{\text{SC}} \cdot \mathcal{L}_{\text{SC}} + \lambda_{\text{round}} \cdot \mathcal{L}_{\text{round}} \tag{6}$$

where $\lambda_{\text{round}}, \lambda_{\text{SC}} \in \mathcal{R}$ are weight values and the definitions of partial loss functions are provided below.

Flow-Matching Loss ($\mathcal{L}_{\text{FM}}$): For a small $d = \epsilon \to 0$, the model is trained to match the ground-truth empirical velocity:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{z_0, z_1, t} \left\| s_\theta(z_t, t, 0) - (z_1 - z_0) \right\|^2 \tag{7}$$

[1] Our code is at https://github.com/jwarczynski/FMShortcut.

Self-Consistency Loss ($\mathcal{L}_{\text{SC}}$): For $d > 0$, the target shortcut vector is constructed recursively, using the trajectory halving rule from Eq. 5:

$$s_{\text{target}} = \frac{1}{2}\left(s_\theta(z_t, t, d) + s_\theta(z_{t+d}, t+d, d)\right) \quad (8)$$

and the loss function is defined as:

$$\mathcal{L}_{\text{SC}} = \mathbb{E}_{y_0, y_1, t, d}\left\| s_\theta(z_t, t, 2d) - s_{\text{target}} \right\|^2 \quad (9)$$

where $y_{t+d} = y_t + d \cdot s_\theta(z_t, t, d)$.

Rounding Loss ($\mathcal{L}_{\text{round}}$) Following (Liu et al., 2024), we also include the rounding loss, which prevents the collapse of learned token embeddings to a single point. We define the loss using negative log-likelihood over a learned embedding matrix Φ:

$$\mathcal{L}_{\text{round}} = -\mathbb{E}_{z_1} \log p_\Phi(w_z \mid z_1) \quad (10)$$

where z_1 is the embedding of the tokenized input-output sequence w_z, and p_Φ is the probability distribution over vocabulary tokens obtained via rounding:

$$p_\Phi(w_z \mid z_1) = \text{softmax}(z_1 \Phi^\top) \quad (11)$$

For pseudocode, see Alg. 1 in the Appendix.

2.4 Text generation

With such defined shortcut model, one-step decoding can be simply realized by setting $d = 1$ and computing:

$$z_1 = z_0 + s_\theta(z_0, 0, 1)$$

3 Experimental setup

We performed multiple computational experiments to assess the usefulness of the proposed shortcut flow-matching models for text generation.

Datasets We experiment with three popular text generation datasets (the task of text paraphrasing): Quora Question Pairs (QQP, DataCanary et al., 2017), PAWS$_{WIKI}$ (Zhang et al., 2019) and ParaSCI (Dong et al., 2021). These datasets were used in related NAR research (Zou et al., 2024; Hu et al., 2024a; Yuan et al., 2024) and their basic characteristics are provided in App. D.

Metrics The quality of the generated texts was mainly assessed using two metrics, used in related research: a standard n-gram–based metric, BLEU (Papineni et al., 2002), and a learned metric, BERTScore (Zhang et al., 2020). In the appendix, the results of other popular metrics are provided: ROUGE-1 and ROUGE-L (Lin, 2004).

Additionally, an LLM-based evaluation was performed using the pre-validated metric Themis (Hu et al., 2024b). It is a reference-free metric that enables evaluation of user-defined quality aspects. We adopted three quality aspects used for the paraphrasing task in the original Themis paper: fluency, semantic similarity, and overall quality. See the definitions in App. C.

The BLEU implementation from the NLTK package was used (Bird and Loper, 2004), Themis evaluation was performed using the code from the original repository (Hu et al., 2024b), and the remaining metrics were computed using the HuggingFace library (Wolf et al., 2020).

Model training The underlying architecture is a Transformer-based encoder which follows BERT-base configuration (Devlin et al., 2019). The optimizer is AdamW (Loshchilov and Hutter, 2019) with an initial learning rate of 10^{-4}. Exponential moving average (EMA) smoothing is applied to model parameters. The models operate at a fixed maximum length (128 tokens for QQP and PAWS$_{WIKI}$, 256 for ParaSCI) and truncate the outputs based on an EOS token. See more details in App. A.

Model extensions We performed additional experiments with the proposed shortcut model coupled with three different, well-established extension strategies for NAR generation.

Self-conditioning was introduced by Strudel et al. (2022) to improve denoising quality by allowing the model to reuse its own intermediate predictions. Instead of predicting the target from the noisy input z_t alone, the model is also conditioned on its earlier estimate of y_t. In practice, this additional input is concatenated to z_t along the feature dimension. During training, the additional input y_t is constructed by performing an additional forward pass without self-conditioning.

Classifier-Free Guidance (CFG, Ho and Salimans, 2022) is a technique to generate instances better aligned with the provided condition. During training, a fraction of conditioning inputs (here: 10%) are randomly replaced by an unconditional placeholder embedding z_t^{uncond}. The model is thus exposed to both conditional and unconditional examples, sharing representations across them. During inference, the conditioning is strengthened by

Model	Steps	QQP BLEU	QQP BERTScore	PAWS_{WIKI} BLEU	PAWS_{WIKI} BERTScore	ParaSCI BLEU	ParaSCI BERTScore
Non-autoregressive flow-matching models							
Flow-matching (baseline, Liu et al., 2024)	1	14.20	0.6170	20.97	0.6465	6.05	0.5454
Shortcut model (ours)	1	19.40	**0.8143**	42.71	0.8634	13.55	**0.7843**
Non-autoregressive diffusion models							
DiffuSeq (Gong et al., 2023)	2000	31.49	-	25.89	0.8447	-	-
SeqDiffuSeq (Yuan et al., 2024)	2000	26.67	-	28.13	-	-	-
Autoregressive models							
Transformer	n/a	29.10	0.8000	42.40	**0.9200**	**27.10**	0.6000
T5	n/a	**33.50**	-	**61.05**	0.8991	26.20	0.5500

Table 1: Text generation performance of the proposed shortcut models compared with standard flow-matching and other related models. For non-autoregressive methods, the number of denoising steps (Steps) is reported.

Model	T.step.	BLEU	BERTS.
Shortcut model	21k	19.4	0.814
+ Self-conditioning	15k	19.3	0.806
+ Classifier-free guid.	23k	19.4	0.806
+ BERT initialization	10k	19.5	0.826
+ embedding matrix init.	46k	18.6	0.773
+ frozen embedding init.	11k	0.0	0.191

Table 2: Results of experiments combining the shortcut model with additional training techniques on the QQP dataset. The number of training steps (T.step.), BLEU and BERTScore (BERTS.) are reported.

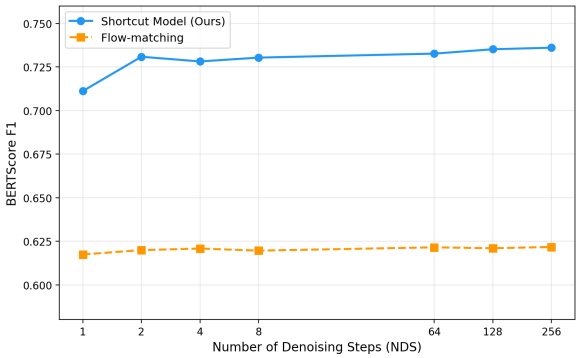

Figure 2: BERTScore results as a function of NDS on the QQP dataset

lowering the probability of tokens predicted by the unconditioned model:

$$s_{\text{CFG}} = (1+w)\cdot s_\theta(z_t, t, d) - w \cdot s_\theta(z_t^{\text{uncond}}, t, d)$$

where w is a hyperparameter controlling the strength of conditional alignment.

BERT initialization While conditioning and guidance determine how information is injected, initialization strongly influences training dynamics. We therefore investigated three strategies of initializing the model with pretrained weights from BERT (Devlin et al., 2019): (1) initialize the entire transformer model (BERT initialization), (2) use only the word embedding matrices (embedding matrix init.) and (3) use pre-initialzed and frozen word embedding matrices (frozen embedding init.).

4 Results

Shortcut models improve one-step generation capabilities The main comparison between the classic flow-matching method and the proposed shortcut models is presented in Tab. 1. For reference, we also report results of several methods from the literature: DiffuSeq (Gong et al., 2023) and SeqDiffuSeq (Yuan et al., 2024), which are diffusion-based models, as well as two autoregressive approaches: a vanilla Transformer (Vaswani et al., 2017) and fine-tuned T5-based models (Raffel et al., 2020) (see details in App. B).

The proposed shortcut models achieve statistically significant improvements on all three datasets (t-test, $p < 0.001$). On two datasets, PAWS_{WIKI} and ParaSCI, our model more than doubles the baseline BLEU score. On QQP, diffusion models achieve the highest performance among non-autoregressive methods. Note that, in contrast to our one-step decoding, the reported results for diffusion models require 2000 denoising steps.

Multi-step generation does not significantly improve shortcut performance We conducted experiments with both the baseline flow-matching model and the shortcut models to evaluate their performance when the number of decoding steps (NDS) exceeds one (NDS > 1). Specifically, we experimented with NDS $\in \{1, 2, 4, 8, 64, 128, 256\}$, focusing on regimes with a small number of model evaluations. Results on the QQP dataset are shown in Fig. 2.

The baseline flow-matching model benefits from an increased number of evaluations, but even its best performance with higher NDS remains below that of the shortcut model. The proposed model's performance peaks at NDS $= 256$ but there is only marginal improvements compared to NDS $= 1$.

BERTScore for the shortcut model shows a slight improvement when the number of steps is increased to NDS $= 2$, and starts to fluctuate while further increasing NDS. The same behavior is visible on BLEU values.

Self-conditioning and pretrained initialization accelerate convergence but yield limited performance gains Results for experiments combining shortcut models with self-conditioning, classifier-free guidance (CFG), and different BERT-based initialization strategies are reported in Tab. 2.

Self-conditioning does not improve performance in the one-step setting; however, models trained with self-conditioning converge faster. This suggests that, while the recursive feedback introduced by self-conditioning aids the multi-step refinement used during training, it does not translate into stronger one-step generation capabilities, where such feedback is not available.

Classifier-free guidance also does not improve one-step generation, resulting even with a small degradation of BERT-Score measure.

Finally, initializing the shortcut model with BERT slightly improves BLEU, though not significantly, and enables fast convergence (within 10k steps). Embedding initialization strategies were not effective – which aligns with prior results for diffusion models (He et al., 2023).

Shortcut models demonstrate improvements also on more demanding LLM-based metrics The results of LLM-based evaluation on Themis (Hu et al., 2024b) on three quality aspects (fluency, semantic similarity and overall quality) are presented in Table 3. The shortcut models provided significant improvements on all aspects over the baseline, but in general the text quality provided by flow-matching models at NDS $= 1$, as measured by Themis, is not very high.

We performed a more detailed analysis, reporting results separately for short sentences (fewer than 9 words) and long sentences (9 words or more). The results are provided in App. F.

Shortcut-based models improved performance across all investigated subsets and evaluation aspects. On QQP, the improvements were larger for short sentences, whereas on PAWS$_{WIKI}$ they were more pronounced for long sentences. On ParaSCI improvements were similar for both subsets, suggesting that the performance of proposed models does not depend strongly on the length of the generated text.

Dataset	Model	Fluency	Sem. Sim.	Over.
QQP	Shortcut	1.63	1.67	1.42
	Baseline	1.13	1.13	1.09
PAWS$_{WIKI}$	Shortcut	2.01	2.07	2.02
	Baseline	1.05	1.04	1.03
ParaSCI	Shortcut	1.70	1.60	1.60
	Baseline	1.00	1.00	1.00

Table 3: LLM-based evaluation (Themis) for Shortcut vs Baseline across datasets (NDS=1) for three aspects: fluency, semantic similarity (Sem. Sim.) and overall quality (Over.).

Limitations

Although one-step decoding is highly efficient at inference time, training the shortcut model introduces additional complexity (estimation of multi-step trajectories) and hyperparameters.

Our experiments are restricted to generation tasks with relatively short-to-medium sequence lengths; it is unclear how well the shortcut mechanism generalizes to other generation settings. The evaluation was also limited to text paraphrasing task. The proposed non-autoregressive approach may still underperform autoregressive models in complex generation settings or fail to generalize robustly across diverse domains and text lengths.

Finally, while shortcut directions are learned to approximate multi-step trajectories, we do not provide theoretical guarantees on stability or error accumulation when extrapolating to very large step sizes, leaving a more formal analysis of shortcut flow behavior for future work.

Acknowledgment

Co-funded by the European Union (ERC, NG-NLG, 101039303). We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2025/018830. This work also used resources of the LINDAT/CLARIAH-CZ Research Infrastructure (Czech Ministry of Education, Youth, and Sports project No. LM2018101).

Ethical Considerations

We acknowledge the use of GitHub Copilot in drafting parts of our code. Some portions of the manuscript were revised using an AI-assisted grammar checker.

References

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Qingxiu Dong, Xiaojun Wan, and Yue Cao. 2021. ParaSCI: A large scientific paraphrase dataset for longer paraphrase generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 424–434, Online. Association for Computational Linguistics.

Betty Fabre, Tanguy Urvoy, Jonathan Chevelu, and Damien Lolive. 2021. Neural-driven search-based paraphrase generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2100–2111, Online. Association for Computational Linguistics.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. 2025. One step diffusion via shortcut models. In *The Thirteenth International Conference on Learning Representations*.

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

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2023. 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)*, pages 4521–4534, Toronto, Canada. Association for Computational Linguistics.

Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *Preprint*, arXiv:2207.12598.

Vincent Hu, Di Wu, Yuki Asano, Pascal Mettes, Basura Fernando, Björn Ommer, and Cees Snoek. 2024a. Flow matching for conditional text generation in a few sampling steps. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 380–392, St. Julian's, Malta. Association for Computational Linguistics.

Xinyu Hu, Li Lin, Mingqi Gao, Xunjian Yin, and Xiaojun Wan. 2024b. Themis: A reference-free NLG evaluation language model with flexibility and interpretability. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15924–15951, Miami, Florida, USA. Association for Computational Linguistics.

Jaehun Jung, Peter West, Liwei Jiang, Faeze Brahman, Ximing Lu, Jillian Fisher, Taylor Sorensen, and Yejin Choi. 2024. Impossible distillation for paraphrasing and summarization: How to make high-quality lemonade out of small, low-quality model. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4439–4454, Mexico City, Mexico. Association for Computational Linguistics.

Sangyun Lee, Beomsu Kim, and Jong Chul Ye. 2023. Minimizing trajectory curvature of ODE-based generative models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 18957–18973. PMLR.

Yifan Li, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Diffusion models for non-autoregressive text generation: A survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 6692–6701. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2023. Flow matching for generative modeling. *Preprint*, arXiv:2210.02747.

Pan Liu, Xiaohua Tian, and Zhouhan Lin. 2024. Enable fast sampling for Seq2Seq text diffusion. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8495–8505, Miami, Florida, USA. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations (ICLR 2019)*. Also available as arXiv:1711.05101.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

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

Jaesung Tae, Hamish Ivison, Sachin Kumar, and Arman Cohan. 2025. TESS 2: A large-scale generalist diffusion language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21171–21188, Vienna, Austria. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Minkai Xu, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. 2025. Energy-based diffusion language models for text generation. In *International Conference on Learning Representations (ICLR)*. Also available as arXiv:2410.21357.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. 2024. 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)*, pages 22–39, Mexico City, Mexico. Association for Computational Linguistics.

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

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaochen Zhu, Georgi Karadzhov, Chenxi Whitehouse, and Andreas Vlachos. 2025. Segment-level diffusion: A framework for controllable long-form generation with diffusion language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4163–4183, Vienna, Austria. Association for Computational Linguistics.

Wei Zou, Ziyuan Zhuang, Xiang Geng, Shujian Huang, Jia Liu, and Jiajun Chen. 2024. Improved paraphrase generation via controllable latent diffusion. *arXiv preprint arXiv:2404.08938*.

Jialong Zuo, Shengpeng Ji, Minghui Fang, Mingze Li, Ziyue Jiang, Xize Cheng, Xiaoda Yang, Chen Feiyang, Xinyu Duan, and Zhou Zhao. 2025. Rhythm controllable and efficient zero-shot voice conversion via shortcut flow matching. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16203–16217, Vienna, Austria. Association for Computational Linguistics.

A Training Details

Model The underlying architecture is a Transformer-based encoder with input and output dimensionality of 768. The model (and tokenization) follows the `bert-base-uncased` configuration, consisting of 12 encoder layers with 12 attention heads per layer. Each attention head operates on a dimension of 64 (768/12), while the feed-forward network within each encoder layer has an intermediate size of 3072. The total model comprises approximately 100 million parameters.

Training details Models are trained with a batch size of 1024 and a maximum of 60,000 update steps. The optimizer is AdamW (Loshchilov and Hutter, 2019) with an initial learning rate of 10^{-4}, linear decay scheduling over 60,000 steps, and weight decay of 0.1. Gradients are clipped at a global norm of 1.0, and exponential moving average (EMA) smoothing with factor 0.99 is applied to model parameters. Dropout is set to 0.1 throughout all layers, including attention and feed-forward components.

The proposed loss function of shortcut models introduces two new hyperparameters $\lambda_{\text{round}}, \lambda_{\text{SC}}$, i.e. weights of the new loss functions. In all our experiments we used the fixed value of 1, so these hyperparameters were not effectively fine-tuned.

B Details on Diffusion and AR Baselines

In Tab. 1, we compare our proposed shortcut models against both diffusion-based and autoregressive baselines reported in the literature. To provide a comprehensive comparison, the results for the autoregressive models are sourced from the following works:

Transformer We report results for the standard Transformer architecture (Vaswani et al., 2017) trained from scratch. For ParaSCI, this corresponds to the baseline established by Dong et al. (2021). For QQP and PAWS$_{WIKI}$, we reference standard benchmarks for the vanilla Transformer generator as reported in Fabre et al. (2021); Zou et al. (2024).

T5-based Models The results in the "T5" row aggregate the best-performing T5 variants identified in recent benchmarks. Specifically:

- **QQP and PAWS$_{WIKI}$**: We report the results of **T5-GPVAE** (Zou et al., 2024), a T5-base model augmented with a Gaussian Prior Variational Autoencoder to improve generation diversity.
- **ParaSCI**: We report the performance of a **T5-Large** model fine-tuned on the ParaBank dataset (T5ParaBank1), as benchmarked by Jung et al. (2024).

C Details on LLM-based evaluation

We adopted three aspect definitions from original Themis paper (Hu et al., 2024b):

- Fluency: Whether the paraphrase is meaningful and grammatical?
- Semantic Similarity: Whether the paraphrase maintains similar semantics to the original text?
- Overall Quality: The paraphrase should not only maintain similar semantics to the original text, but also possess lexical or syntactic differences from the original text, with fluent and coherent content

D Dataset Specifications

We evaluated our models on three distinct paraphrase datasets: PAWS$_{WIKI}$, QQP, and ParaSCI. The statistics for the training, validation, and test splits used in our experiments are summarized in Table 4.

Preprocessing and Filtering

- **PAWS$_{WIKI}$**: We sourced the dataset from the official Google Research repository[2]. Since the original dataset includes both paraphrase and non-paraphrase pairs for classification tasks, we filtered the dataset to retain *only* the positive paraphrase pairs for the generation task.
- **QQP (Quora Question Pairs)**: We utilized the version provided by the FMSeq repository[3], where the dataset consists exclusively of paraphrase pairs.
- **ParaSCI**: We used the ACL split from the Hugging Face `HHousen/ParaSCI` repository, which focuses on scientific paper paraphrases. No additional filtering was required.

Table 4: Dataset statistics. Domains: PAWS (Wikipedia), QQP (Quora), ParaSCI (Scientific Papers).

Dataset	Train	Valid	Test	Total
PAWS$_{WIKI}$	21,829	3,539	3,536	28,904
QQP	144,715	2,048	100	146,863
ParaSCI	338,717	6,433	4,894	350,044

E Shortcut Model Training Algorithm

The pseudocode of the model training is provided in Alg. 1.

F Additional results

The comparison of shortcut models with baseline flow-matching models is presented in Table 5.

The results of Themis (Hu et al., 2024b) with sentence-length split are presented in Table 6. The results of Themis were estimated on a random sample of 500 examples (100 for QQP) using the `max_samples` flag provided by Themis package. All short sentences (below 9 words) were evaluated for all datasets, and the performance for long

[2] https://github.com/google-research-datasets/paws

[3] https://github.com/Peacer68/FMSeq

Algorithm 1 Training procedure for Shortcut Flow-Matching Model

Require: Dataset $\mathcal{D}$, Model s_θ, Batch size B, Loss weights $\lambda_{SC}, \lambda_{round}$
Require: Max step size d_{max}
Initialize model parameters θ
while not converged **do**
  Sample batch of target texts $y_1 \sim \mathcal{D}$
  Sample source texts x (if conditional)
  $z_1 \leftarrow \text{Embed}(x, y_1)$ ▷ Construct target embedding
  Sample $z_0 \sim \mathcal{N}(0, I)$
  Sample time steps $t \sim \mathcal{U}[0, 1]$
  1. Flow Interpolation
  $z_t \leftarrow (1-t)z_0 + tz_1$ ▷ Linear interpolation
  $v_{target} \leftarrow z_1 - z_0$ ▷ Ground truth velocity
  2. Flow-Matching Loss ($d = 0$)
  $\hat{v}_0 \leftarrow s_\theta(z_t, t, 0)$
  $\mathcal{L}_{FM} \leftarrow ||\hat{v}_0 - v_{target}||^2$
  3. Self-Consistency Loss ($d > 0$)
  Sample step sizes $d \sim \mathcal{U}[0, d_{max}]$ s.t. $t + 2d \leq 1$
  $v_{step1} \leftarrow s_\theta(z_t, t, d)$ ▷ First small step
  $z_{next} \leftarrow z_t + d \cdot v_{step1}$ ▷ Intermediate state
  $v_{step2} \leftarrow s_\theta(z_{next}, t + d, d)$ ▷ Second small step
  $v_{sc\_target} \leftarrow \frac{1}{2}(v_{step1} + v_{step2})$ ▷ Target for large step (Eq. 3)
  $\hat{v}_{2d} \leftarrow s_\theta(z_t, t, 2d)$ ▷ Predict large step directly
  $\mathcal{L}_{SC} \leftarrow ||\hat{v}_{2d} - v_{sc\_target}||^2$
  4. Rounding Loss
  $p_\Phi(w|z_1) \leftarrow \text{Softmax}(z_1\Phi^\top)$ ▷ Project to vocab space
  $\mathcal{L}_{round} \leftarrow -\log p_\Phi(w_{true}|z_1)$ ▷ NLL of true tokens (Eq. 5)
  5. Optimization
  $\mathcal{L}_{total} \leftarrow \mathcal{L}_{FM} + \lambda_{SC}\mathcal{L}_{SC} + \lambda_{round}\mathcal{L}_{round}$
  Update θ using $\nabla_\theta \mathcal{L}_{total}$
end while

sentences (9 words or more) was estimated on 200 random sentences that meet the length constraint.

Dataset	Model	BLEU	ROUGE-1	ROUGE-L	BERTScore
QQP	Flow-matching (baseline)	14.20	0.404	0.380	0.6170
	Shortcut model	19.40	0.511	0.485	0.8143
PAWS_{WIKI}	Flow-matching (baseline)	20.97	0.578	0.518	0.6465
	Shortcut model	42.71	0.812	0.724	0.8634
ParaSCI	Flow-matching (baseline)	6.05	0.270	0.254	0.5454
	Shortcut model	13.55	0.509	0.486	0.7843

Table 5: Comparison of shortcut and baseline flow-matching models across three datasets (NDS=1 for all settings).

Dataset	Test examples	Model	Fluency	Semantic Similarity	Overall Quality
QQP	All (n=100)	Shortcut	1.63	1.67	1.42
		Baseline	1.13	1.13	1.09
	Short $<$9 words (n=36)	Shortcut	2.31	2.42	1.86
		Baseline	1.14	1.17	1.11
	Long $\geq$9 words (n=64)	Shortcut	1.25	1.25	1.19
		Baseline	1.13	1.11	1.08
PAWS_{WIKI}	All (n=500)	Shortcut	2.01	2.07	2.02
		Baseline	1.05	1.04	1.03
	Short $<$9 words (n=32)	Shortcut	1.56	1.67	1.56
		Baseline	1.25	1.31	1.00
	Long $\geq$9 words (n=200)	Shortcut	2.11	2.34	2.32
		Baseline	1.03	1.02	1.03
ParaSCI	All (n=500)	Shortcut	1.70	1.60	1.60
		Baseline	1.00	1.00	1.00
	Short $<$9 words (n=191)	Shortcut	1.74	1.65	1.64
		Baseline	1.02	1.03	1.02
	Long $\geq$9 words (n=200)	Shortcut	1.72	1.63	1.56
		Baseline	1.01	1.01	1.01

Table 6: LLM-based evaluation (Themis, Hu et al., 2024b) for shortcut vs baseline models across datasets and sentence length splits (NDS=1). Sentences are divided into short (fewer than 9 words) and long (9 words or more), the size of a random sample used to estimate the result is given in parenthesis