Diving into the Decoding Space of Non-Autoregressive Models via Lexically Constrained Search

Chenyang Huang, Osmar R. Zaïane
Dept. of Computing Science, Alberta Machine Intelligence Institute (Amii), University of Alberta
chenyangh@ualberta.ca zaiane@ualberta.ca

Abstract

Non-autoregressive (NAR) models have been mainly developed to improve decoding efficiency. Lately, they have also shown potential in controlled text generation tasks. In this work, we investigate the decoding space of NAR models through lexically constrained machine translation tasks, and develop a search-based decoding algorithm named LexMAP for the Directed Acyclic Transformer (DAT). LexMAP achieves full constraint coverage and competitive BLEU, but trails the autoregressive (AR) model with Grid Beam Search (GBS) on COMET and MetricX. Our analysis reveals several interesting properties of DAT-based NAR decoding: 1) LexMAP is less affected by maximum a posteriori (MAP) degradation than AR with GBS; 2) AR beam search exhibits strong positional bias, in which the candidates only diverge near the end of the sequence; 3) LexMAP accumulates more DAT probability mass, with a sharper-distribution caveat.[1]

1 Introduction

Language modeling has been a cornerstone of the recent blooming of artificial intelligence, demonstrating remarkable capabilities in various advanced tasks such as math, coding, and puzzle solving (Vaswani et al., 2017; Devlin et al., 2019; OpenAI, 2023; DeepSeek-AI, 2024). The *de facto* approach for language modeling is to use an autoregressive (AR) model, which factorizes the sequence-level probability into a product of conditional probabilities, and each condition includes all the previous tokens.

Despite being widely used, AR models have long been criticized for several limitations, including exposure bias and label bias (Lin et al., 2021), and slow decoding speed (Sun et al., 2019; Stern et al., 2018). Consequently, alternative approaches like diffusion language models and non-autoregressive (NAR) models have been proposed.

In particular, NAR models make independent predictions, which allows for parallel decoding, significantly improving the decoding speed. However, this loss of dependency also leads to challenges in generating coherent responses, which has been a main research problem for the NAR research community. Among the advances, the Directed Acyclic Transformer (DAT) (Huang et al., 2022) stands out; it predicts extra tokens and also predicts links to select which tokens to keep in the final output, effectively improving the modeling capacity while maintaining efficiency.

In addition to efficiency, NAR models also offer unique advantages in controlled generation tasks (Liu et al., 2022a,b; Huang et al., 2024). Because NAR models make independent predictions, they enable large controlled-generation problems to be decomposed into smaller, independent subproblems, which can often be solved efficiently via dynamic programming. Despite these promising results, the existing research primarily focuses on the length-control tasks, where the autoregressive (AR) models are not able to perform efficient search. As a result, there is a lack of understanding of how NAR models behave under search-based decoding, compared with their autoregressive counterparts.

In this research, we investigate the decoding space of NAR models through the lens of lexically constrained text generation tasks, which aim to find the most probable generation that contains a list of pre-specified words. Compared with length-control tasks, there are existing search-based decoding algorithms for autoregressive (AR) models (Hokamp and Liu, 2017; Post and Vilar, 2018; Hu et al., 2019). Therefore, by developing an analogous search-based decoding algorithm for NAR models, we can compare AR and NAR decoding under the same lexically constrained setting. Specifically, we

[1]The code, model, dataset, and output of this work are publicly available at https://github.com/chenyangh/DAT-lex-control-decoding.git

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 898–906
July 2-7, 2026

propose a novel dynamic programming (DP) algorithm for lexically constrained decoding based on the Directed Acyclic Transformer (DAT) (Huang et al., 2022). The general idea is to track the number of satisfied constraints at each of DAT's prediction steps and apply beam search to store the top sequences. This beam search algorithm can be efficiently implemented with dynamic programming, as the predictions at each step are independent.

Our experiments generally follow the settings for lexically constrained machine translation (Hokamp and Liu, 2017; Susanto et al., 2020). The language models are trained on the WMT'14 English–German training set (Bojar et al., 2014) and evaluated on three test sets built by filtering the WMT'14 and WMT'17 En–De test sets (Bojar et al., 2014, 2017) with Wiktionary and IATE terminology lists (European Union, 2007). We also add Zh–En to test different word order. We compare our proposed method with a strong AR baseline: Grid Beam Search (GBS) (Hokamp and Liu, 2017). The results show that our proposed DAT-based method achieves full coverage and competitive BLEU, but lower COMET/MetricX than AR with GBS. More importantly, we thoroughly analyze the decoding space of DAT-based NAR models and have the following key findings: 1) LexMAP is less affected by maximum a posteriori (MAP) degradation than AR with GBS (Meister et al., 2020); 2) AR beam search exhibits strong positional bias, in which the candidates tend to diverge near the end of the sequence; 3) LexMAP accumulates more model probability mass, with a sharper-distribution caveat.

2 Related Work

Existing attempts at lexically controlled translation have built upon both autoregressive (AR) and iterative non-autoregressive (NAR) models. Specifically, autoregressive models predict the next word based on the previously generated words, and exact lexical control is achieved by augmenting the standard decoding algorithm—beam search—with additional states, which progressively force that the lexical constraints are covered (Hokamp and Liu, 2017; Post and Vilar, 2018; Hu et al., 2019). These approaches are search-based, aiming to find the best translation in the AR model's decoding space that satisfies the given lexical constraints.

On the other hand, existing NAR approaches rely on iterative editing: the model starts with the required lexical constraints and then progressively completes the generation (Susanto et al., 2020; Zeng et al., 2022). Such decoding methods are greedy, where the model only predicts the most probable words or word spans at each iteration, and therefore are not within the scope of search-based methods. They are relevant NAR comparison points, but they do not provide insights into the decoding space of NAR models, e.g., what kind of sequences NAR and AR models prefer, and how different they are when both are under constraints.

Our work is closely related to controlled text generation with non-autoregressive models. Previously, Liu et al. (2022a) and Liu et al. (2022b) proposed NAR models with dynamic programming algorithms for length-control text generation, where the Connectionist Temporal Classification (CTC, Graves et al., 2006) model is used as the backbone. Their insight is that searching over NAR models' decoding spaces can be efficiently coupled with dynamic programming, leading to better performance. Later, Huang et al. (2024) verify this hypothesis with the Directed Acyclic Transformer (DAT) (Huang et al., 2022), which is a stronger NAR model. While these works achieve better generation quality, there is still a lack of studies on how NAR and AR models behave under search-based decoding. Therefore, we aim to fill this gap by investigating the decoding space of NAR models via lexically constrained generation.

3 Methodology

3.1 DAT Model

The Directed Acyclic Transformer (DAT) (Huang et al., 2022) is a non-autoregressive model that generates words in parallel, allowing for efficient decoding. DAT consists of two main components: word predictions and link predictions.

Given an input sequence $\mathbf{x} = (x_1, \ldots, x_{T_\mathrm{x}})$ and a target sequence $\mathbf{y} = (y_1, \ldots, y_{T_\mathrm{y}})$, DAT makes S prediction steps, where $S \geq T_\mathrm{y}$, to accommodate the target sequence. The target words are collected from a selection of S steps, denoted by $\boldsymbol{a} = (a_1, \cdots, a_{T_\mathrm{y}})$, where $1 = a_1 < a_2 < \cdots < a_{T_\mathrm{y}} = S$. Further, DAT predicts the links for every step a_s, modeling the probability distribution of the next step's position, given by $p^{(s)}_{\text{link}}(\cdot \mid \mathbf{x}), \text{for } s \in \{1, \cdots, S\}$. For each step s, a word prediction is also made, given by $p^{(s)}_{\text{word}}(\cdot \mid \mathbf{x}), \text{for } s \in \{1, \cdots, S\}$. For simplicity, we denote the word probability of token v at step s as $w_{s,v} = p^{(s)}_{\text{word}}(v \mid \mathbf{x})$ and the link probability

from step i to step j as $l_{i,j} = p^{(i)}_{\text{link}}(j \mid \mathbf{x}), i < j$.

For the target sequence $\mathbf{y}$, the probability is given by marginalizing over all possible selections $\boldsymbol{a}$, and each selection's probability is computed by multiplying the link probabilities and word probabilities along the selected steps. This is given by

$$p(\mathbf{y} \mid \mathbf{x}) = \sum_{\boldsymbol{a} \in \Gamma_{T_\mathrm{y},S}} \prod_{t=2}^{T_\mathrm{y}} l_{a_{t-1},a_t} \prod_{t=1}^{T_\mathrm{y}} w_{a_t,y_t} \quad (1)$$

where $\Gamma_{T_\mathrm{y},S}$ is the set of all possible selections of T_y steps from S steps. Because predictions at each step are independent, this marginalizing can be efficiently computed with dynamic programming (Huang et al., 2022).

3.2 Lexical Control

For lexical-control text generation, a list of constraints $\mathbf{c} = (\mathrm{c}_1, \ldots, \mathrm{c}_M)$, where $\mathrm{c}_m \in \mathcal{V}$ and M is the total number of constraints, is satisfied by a target sequence $\mathbf{y}$ if (i) each word in $\mathbf{c}$ appears in $\mathbf{y}$ in the same order, and (ii) word c_i and word c_{i+1} must be adjacent in $\mathbf{y}$ if c_i is a continuation (or non-terminating) token; we denote the set of continuation tokens by $\mathcal{V}^{(\text{cont})}$.[2]

Objective. We denote the set of all target sequences that satisfy the constraints $\mathbf{c}$ as $\mathcal{V}(\mathbf{c})$. The lexically controlled decoding objective is to find the target sequence $\mathbf{y}^\star$ that maximises the probability of generating $\mathbf{y}$ under the constraints $\mathbf{c}$. Integrating the DAT probability from Eq. (1), the objective can be rewritten as

$$\mathbf{y}^\star = \arg\max_{\mathbf{y} \in \mathcal{V}(\mathbf{c})} \sum_{s=T_\mathrm{y}}^{S} \sum_{\boldsymbol{a} \in \Gamma_{T_\mathrm{y},s}} \prod_{t=2}^{T_\mathrm{y}} l_{a_{t-1},a_t} \prod_{t=1}^{T_\mathrm{y}} w_{a_t,\mathrm{y}_t} \quad (2)$$

LexMAP. However, finding the exact solution to the objective in Eq. (2) is non-trivial; we develop an approximate algorithm based on beam search and apply dynamic programming to find high-probability sequences efficiently.

[2]In practice, words in human languages are segmented into subwords, which results in more shared word segments and thus reduces the demand for a large vocabulary size. In this work, we use the Byte Pair Encoding (BPE) (Sennrich et al., 2016) method, where a special marker "@@" is added to the end of each subword except the last one. For example, the word "playing" may be segmented into "play@@" and "ing", and we denote words like "play@@" as *continuation tokens*, given by $\mathcal{V}^{(\text{cont})} \subseteq \mathcal{V}$.

This algorithm is named LexMAP (Lexically Constrained Maximum A Posteriori) decoding for DAT. Although LexMAP is novel, it is not the sole focus of this work, which is to investigate the decoding space of NAR models. Therefore, we refer readers to Appendix A for more details.

4 Experiments

4.1 Setup

Our experimental settings generally follow the work by Susanto et al. (2020), which are also adopted by Lee et al. (2021) and Zeng et al. (2022). The exact details are organized in Appendix A.2.

Datasets. We evaluate lexical control mainly on three En–De benchmarks constructed from the WMT'14 and WMT'17 En–De test sets with Wiktionary and IATE terminology constraints. To test whether the findings transfer beyond language pairs with relatively similar word order, we additionally curate a Zh–En benchmark from WMT'17, since Chinese and English differ substantially in word order and alignment patterns. We construct this benchmark by matching public English terminology resources against the English reference side and using the matched target terms as lexical constraints. Details of the dataset construction are provided in Appendix A.2.

Metrics. For lexical control, we report term coverage (%), the percentage of required terms that appear in the system output. To evaluate translation quality, we report BLEU (Papineni et al., 2002), COMET (Rei et al., 2020), MetricX (Juraska et al., 2023), and perplexity (PPL). COMET scores in the result tables are reference-based evaluation scores, while reference-free COMET is used only for beam reranking to avoid using the reference during candidate selection. We report both COMET and MetricX to reduce potential bias introduced by any single metric. For the added Zh–En table, we report the same metrics with $K = 5$ decoding.

Baselines. We mainly compare our decoding method against the autoregressive baseline: Grid Beam Search (GBS, Hokamp and Liu, 2017). Both GBS and LexMAP are beam search-based methods, and they expand each beam into multiple beams to ensure that all lexical constraints are satisfied.

4.2 Results and Analysis

Main Results. Table 1 presents the main En–De results of our proposed DAT-based LexMAP method, compared with the autoregressive Transformer with

#	Method	Coverage↑	BLEU↑	COMET↑	MetricX↓	PPL↓
	n17-Dinu-IATE					
1	Transformer	83.6%	29.55	**0.9030**	2.566	1557.8
2	w/ GBS	**100.0%**	30.08	0.8997	**2.507**	1239.7
3	DAT	81.2%	29.87	0.8771	3.033	778.1
4	w/ LexMAP	**100.0%**	**30.11**	0.8692	3.161	**769.6**
	n14-Wikt					
5	Transformer	81.2%	26.72	**0.9265**	**1.858**	**707.2**
6	w/ GBS	**100.0%**	26.55	0.9254	1.876	709.9
7	DAT	81.6%	27.43	0.9033	2.327	797.6
8	w/ LexMAP	**100.0%**	**27.45**	0.9020	2.339	798.2
	n17-Dinu-Wikt					
9	Transformer	83.5%	30.91	**0.9129**	**2.222**	674.9
10	w/ GBS	**100.0%**	31.00	0.9079	2.296	**620.9**
11	DAT	83.3%	31.10	0.8946	2.641	706.7
12	w/ LexMAP	**100.0%**	**31.31**	0.8800	2.914	704.5

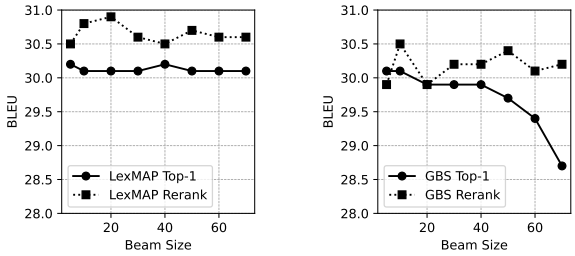
Table 1: Translation quality and constraint satisfaction results on lexically constrained machine translation. Best results within each dataset are in **bold**. The beam size is set to 5 for all methods.

#	Method	Coverage↑	BLEU↑	COMET↑	MetricX↓	PPL↓
	Zh–En					
1	Transformer	62.5%	28.60	**0.7911**	6.180	1510.7
2	w/ GBS	**100.0%**	28.57	0.7873	**6.052**	1458.7
3	DAT	60.0%	**28.61**	0.7709	6.982	1533.8
4	w/ LexMAP	**100.0%**	27.86	0.7632	7.194	**1299.7**

Table 2: $K = 5$ decoding results on the Zh–En lexical-control set. Best results are in **bold**. BLEU and coverage are recomputed from the decoded outputs; COMET, MetricX, and PPL follow the same evaluation setup as Table 1.

GBS decoding. Overall, our LexMAP method achieves 100% constraint coverage and competitive BLEU, but lower COMET/MetricX than AR with GBS. Notably, the DAT-based methods yield higher BLEU scores than the AR baselines yet lower COMET and MetricX scores. This is consistent with a unique property of NAR models, which tend to focus on word-level accuracy, but the overall sentence fluency may be compromised due to the independent predictions. This phenomenon has also been observed in previous NAR studies (Ghazvininejad et al., 2019; Gu and Kong, 2021).

On Zh–En, Table 2 shows a similar constraint-satisfaction pattern but a clearer quality gap for DAT-based decoding. Both GBS and LexMAP improve coverage to 100%, confirming that the search algorithms enforce the target-side constraints. For the autoregressive model, GBS keeps BLEU close to the unconstrained Transformer and improves MetricX. For DAT, LexMAP also reaches full coverage, but its BLEU, COMET, and MetricX scores are lower than the unconstrained DAT output, although it obtains lower PPL. These results suggest that Zh–En is a more difficult lexical-control setting



(a) DAT (b) GBS

Figure 1: Comparison of beam reranking methods.

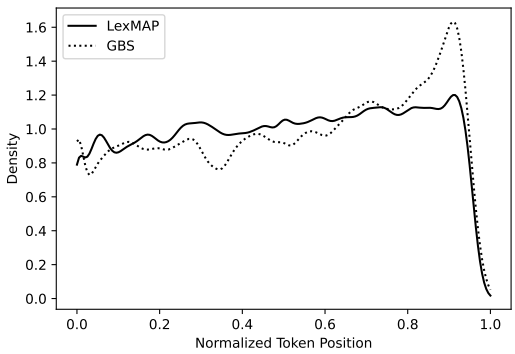


Figure 2: Positional-diversity comparison in beam search. The beam size is set to 30 for both methods.

for DAT-based search, consistent with our motivation for testing a language pair with different word order and less monotonic alignment.

Quality of Beam Search. To evaluate the quality of beam search, we increase the beam sizes from 5 to 70 and evaluate the top-1 beam sequence as well as the best sequence after reranking. Reranking uses reference-free COMET; table COMET scores are reference-based. As shown in Figure 1, we make two observations. First, the autoregressive GBS method suffers from the known beam search degradation issue (Meister et al., 2020), where larger beam sizes lead to worse top-1 sequence quality. On the other hand, our DAT-based LexMAP's performance remains stable across different beam sizes. Second, reranking with reference-free COMET grants more improvements for our LexMAP method than GBS (0.8 BLEU vs. 0.4 BLEU), suggesting that LexMAP keeps useful alternative DAT candidates in the beam.

Decoding Speed. Table 3 reports wall-clock decoding time as the beam size changes. GBS is faster at small beam sizes, but its runtime grows quickly with the beam size. LexMAP has larger overhead when the beam is small, yet scales more slowly and becomes faster from beam size 20 onward. Since

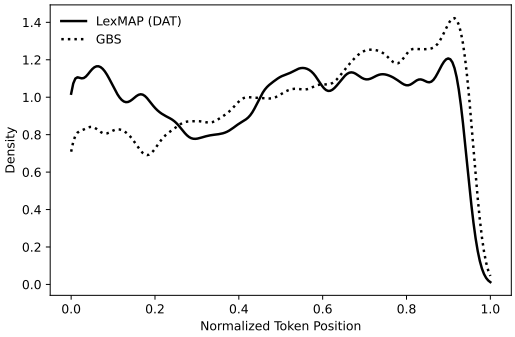
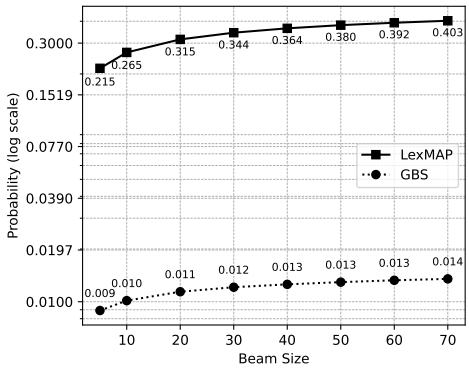


Figure 3: Positional-diversity comparison on the Zh–En lexical-control set. The beam size is set to $K = 30$ for both GBS and LexMAP.



Method	Beam	Mean Top-1	Mean Except Top-1
LexMAP	30	-0.200579	-0.297723
LexMAP	5	-0.201013	-0.246168
GBS	30	-14.485932	-15.885086
GBS	5	-14.585217	-15.571707

Figure 4: Accumulated beam probabilities across beam sizes (top) and per-token log-probability statistics on n17-Dinu-IATE (bottom). The table groups the top-1 hypothesis separately from the remaining beam candidates.

the two systems use different implementations, we treat these numbers as evidence about beam-size scaling rather than a definitive end-to-end speedup comparison.

Positional Bias. Since autoregressive models generate words from left to right, and the follow-up search depends on the previously generated context on the left, we hypothesize that autoregressive beam search exhibits a bias in terms of positions. To verify this, we compare every pair of beam candidates and record the token positions where they begin to diverge. Specifically, we normalize these positions by the sequence length, aggregate them across samples, and plot the resulting distribution.

As seen in Figures 2 and 3, the beam items only begin to diverge close to the end of the sequence

Method	Beam Size				
	5	10	20	40	60
GBS	16.52	35.32	120.76	271.32	460.84
LexMAP	27.25	50.55	57.99	64.13	76.60

Table 3: Wall-clock decoding time in seconds per sentence on n17-Dinu-IATE with batch size 1.

for GBS, indicating that the autoregressive beam search algorithm tends to keep candidates of the same prefix. This also explains why reranking helps less for GBS, as it does not explore changes across different positions. In contrast, LexMAP is more evenly distributed across positions in both comparisons.

Accumulated Probabilities and Distribution Sharpness. We further analyze the accumulated probabilities of the top beam candidates with distribution sharpness. Specifically, we sum up the probabilities of all beam candidates and plot them against different beam sizes, and report per-token log-probability statistics. As shown in Figure 4, our DAT-based LexMAP consistently accumulates higher model probability mass than AR with GBS across different beam sizes. Because AR and DAT use different factorizations and DAT has a sharper beam distribution, we interpret this as concentration within the DAT search space, not as a universal AR–NAR search-quality comparison.

5 Conclusion

In this work, we investigate the decoding space of non-autoregressive (NAR) models via lexically constrained generation. We develop a novel search-based decoding algorithm named LexMAP based on the Directed Acyclic Transformer (DAT). Our experimental results show that our proposed method achieves reliable constraint satisfaction and competitive BLEU, but lower neural-metric scores than autoregressive Transformer with GBS decoding. More importantly, we analyze the decoding space of DAT-based NAR models and reveal several interesting properties, including a smaller beam search degradation issue, less positional bias, and higher accumulated beam probabilities under the DAT distribution. These findings shed light on the unique features of NAR-based text generation and highlight future NAR research.

Limitations

This short paper mainly focuses on investigating the decoding space of DAT-based non-autoregressive (NAR) models via lexically constrained generation. Therefore, we limit our baseline method to the autoregressive (AR) Grid Beam Search (GBS) method. Thus, findings should be read as DAT-versus-AR results rather than claims about all NAR architectures. Unlike most NAR methods, we report runtime in Table 3 but avoid a definitive speedup claim, because the implementations of GBS and our LexMAP are not comparable: GBS is implemented in Python, and our LexMAP is optimized with PyTorch, allowing efficient parallel computation on GPUs. Therefore, reporting a direct speedup comparison may be misleading. The study is also limited to MT and one NAR architecture.

References

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214.

DeepSeek-AI. 2024. Deepseek-V2: A strong, economical, and efficient mixture-of-experts language model.

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*, pages 4171–4186.

European Union. 2007. Iate: Interactive terminology for europe.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6112–6121.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine learning*, pages 369–376.

Jiatao Gu and Xiang Kong. 2021. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 120–133.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved lexically constrained decoding for translation and monolingual rewriting. 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 839–850.

Chenyang Huang, Hao Zhou, Cameron Jen, Kangjie Zheng, Osmar Zaiane, and Lili Mou. 2024. A decoding algorithm for length-control summarization based on directed acyclic transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11572–11583.

Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. 2022. Directed acyclic transformer for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 9410–9428.

Juraj Juraska, Mara Finkelstein, Daniel Deutsch, Aditya Siddhant, Mehdi Mirzazadeh, and Markus Freitag. 2023. MetricX-23: The Google Submission to the WMT 2023 Metrics Shared Task. In *Proceedings of the Eighth Conference on Machine Translation*, pages 756–767.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Gyubok Lee, Seongjun Yang, and Edward Choi. 2021. Improving lexically constrained neural machine translation with source-conditioned masked span prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 743–753.

Chu-Cheng Lin, Aaron Jaech, Xin Li, Matthew R. Gormley, and Jason Eisner. 2021. Limitations of autoregressive models and their alternatives. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5147–5173, Online.

Puyuan Liu, Chenyang Huang, and Lili Mou. 2022a. Learning non-autoregressive models from search for unsupervised sentence summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7916–7929.

Puyuan Liu, Xiang Zhang, and Lili Mou. 2022b. A character-level length-control algorithm for non-autoregressive sentence summarization. *Advances in Neural Information Processing Systems*, pages 29101–29112.

Clara Meister, Ryan Cotterell, and Tim Vieira. 2020. If beam search is the answer, what was the question? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2185.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

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.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, page 10107–10116.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhi-Hong Deng. 2019. Fast structured decoding for sequence models. In *Advances in Neural Information Processing Systems*, pages 3011–3020.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. 2020. Lexically constrained neural machine translation with Levenshtein transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3536–3543.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Chun Zeng, Jiangjie Chen, Tianyi Zhuang, Rui Xu, Hao Yang, Qin Ying, Shimin Tao, and Yanghua Xiao. 2022. Neighbors are not strangers: Improving non-autoregressive translation under low-frequency lexical constraints. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5777–5790.

A Appendix A

A.1 Decoding Algorithm for LexMAP

We propose an approximate algorithm for LexMAP. The idea is to use dynamic programming while tracking, at each DP step, the number of lexical constraints satisfied.

Specifically, let $\mathcal{B}_{t,s,n}$ denote the set of top-K length-t sequences that have satisfied the first n lexical constraints at DAT's prediction step s. Furthermore, let $\mathcal{V}^t(\mathbf{c}_{1:n})$ denote the set of all length-t sequences that satisfy the first n lexical constraints, $\mathbf{c}_{1:n} = (\mathrm{c}_1, \ldots, \mathrm{c}_n)$, where $0 \leq n \leq M$ and M is the total number of lexical constraints.

We denote $\mathcal{B}_{t,s,n}$ and its elements by

$$\mathcal{B}_{t,s,n} = \left\{\mathbf{b}^{(k)}\right\}_{k=1}^{K} \tag{3}$$

where $\mathbf{b}^{(k)} \in \mathcal{V}^t(\mathbf{c}_{1:n})$ is the kth highest scored sequence found during the DP recursion.

Additionally, we store the probability of sequence $\mathbf{b}$ generated at DAT's prediction step s with n constraints satisfied; we denote it by $r_{s,n}(\mathbf{b})$.[3]

The DP recursion iterates along three dimensions. The innermost loop is the number of constraints satisfied. For each length t and prediction step s, we progressively look for the best sequences that satisfy n lexical constraints for $0 \leq n \leq \min(t, M)$, where $n = 0$ is the base scenario where no constraints are satisfied.

The iterations over length t and prediction step s follow the DAT dynamic-programming recursion, where we iterate s for $t \leq s \leq S$, for the same length t, and then progressively increase the length t from 1 to a predefined maximum length.

The core change in LexMAP is the iteration over the number of satisfied constraints. For each $\mathcal{B}_{t,s,n}$, there are two possible ways to form new sequences:

[3]This is different from the marginalized DAT probability in Eq. (1), which sums over possible paths.

free word generation and *constrained word generation*.

Free Word Generation. Free word predictions allow appending any word from the vocabulary $\mathcal{V}$ to previously generated sequences. However, there are two conditions to apply free word generation:

1. The number of satisfied constraints n is not increased,
2. If $n > 0$, the nth constrained word cannot be in $\mathcal{V}^{(\text{cont})}$,[4] such that subword-composed words are not broken apart.

If the conditions are not satisfied, we set the set of freely generated sequences to $\mathcal{B}^{(\text{free})}_{t,s,n} = \emptyset$. Otherwise, we obtain $\mathcal{B}^{(\text{free})}_{t,s,n}$ by considering all shorter-length sequences in each $\mathcal{B}_{t-1,s',n}$, for $t-1 \leq s' < s$. This is given by

$$\mathcal{B}^{(\text{free})}_{t,s,n} = \Big\{ \mathbf{b} \oplus \mathrm{v} \;\Big|\; \mathbf{b} \in \bigcup_{s'=t-1}^{s-1} \mathcal{B}_{t-1,s',n},\ \mathrm{v} \in \mathcal{V} \Big\}, \tag{4}$$

where $\oplus$ denotes sequence concatenation. For search efficiency, we only consider the top-V most probable predicted words at step s. The scores of the new sequences are computed by accumulating their scores from the previous steps:

$$\begin{aligned} r^{(\text{free})}_{s,n}(\mathbf{b} \oplus \mathrm{v}) \;&= w_{s,v} \sum_{s'=t-1}^{s-1} \mathbb{1}\big(\mathbf{b} \in \mathcal{B}_{t-1,s',n}\big) \\ &\quad \cdot\ r_{s',n}(\mathbf{b})\ l_{s',s} \end{aligned} \tag{5}$$

where $\mathbb{1}$ is the indicator function that returns 1 if $\mathbf{b} \in \mathcal{B}_{t-1,s',n}$ and 0 otherwise.

Constrained Word Generation. When the number of satisfied constraints is increased, we need to ensure that the new word is one of the constrained words. Specifically, for $0 < n \leq \min(t, M)$, we construct a new set of sequences $\mathcal{B}^{(\text{const})}_{t,s,n}$ by checking all shorter-length less-constrained sequences within each $\mathcal{B}_{t-1,s',n-1}$, for $t-1 \leq s' < s$, given by

$$\mathcal{B}^{(\text{const})}_{t,s,n} = \Big\{ \mathbf{b} \oplus \mathrm{c}_n \;\Big|\; \mathbf{b} \in \bigcup_{s'=t-1}^{s-1} \mathcal{B}_{t-1,s',n-1} \Big\}, \tag{6}$$

[4] In other words, $\mathrm{c}_n \notin \mathcal{V}^{(\text{cont})}$, where $\mathcal{V}^{(\text{cont})}$ is the set of all continuation tokens, as described in Section 3.2.

where $\oplus$ denotes sequence concatenation, and the scores of the new sequences are given by

$$\begin{aligned} r^{(\text{const})}_{s,n}(\mathbf{b} \oplus \mathrm{c}_n) \;&= w_{s,\mathrm{c}_n} \sum_{s'=t-1}^{s-1} \mathbb{1}\big(\mathbf{b} \in \mathcal{B}_{t-1,s',n-1}\big) \\ &\quad \cdot\ r_{s',n-1}(\mathbf{b})\ l_{s',s} \end{aligned} \tag{7}$$

The new set of sequences $\mathcal{B}_{t,s,n}$ is obtained by merging the two sets $\mathcal{B}^{(\text{free})}_{t,s,n}$ and $\mathcal{B}^{(\text{const})}_{t,s,n}$ and keeping the top-K sequences by score, given by

$$\mathcal{B}_{t,s,n} = \text{Top-}K\ \mathcal{B}^{(\text{free})}_{t,s,n} \cup \mathcal{B}^{(\text{const})}_{t,s,n} \tag{8}$$

where the ranking is according to the scores of both free predictions and constrained predictions, given by

$$r_{s,n}(\mathbf{b}) = r^{(\text{free})}_{s,n}(\mathbf{b}) + r^{(\text{const})}_{s,n}(\mathbf{b}) \tag{9}$$

Here, both $r^{(\text{free})}_{s,n}(\mathbf{b})$ and $r^{(\text{const})}_{s,n}(\mathbf{b})$ are set to 0 if $\mathbf{b}$ is not in the corresponding set.

To obtain the sequences containing all the constraints, we repeat the DP process to obtain each $\mathcal{B}_{t,s,n}$ for $n = 0, \ldots, M$, $s = t, \ldots, S$, and $t = 1, \ldots, T_{\max}$, where $T_{\max}$ is the predefined maximum output length. We then collect all sequences satisfying all M constraints at DAT's final step S, given by

$$\mathcal{B}^{(\text{final})} = \bigcup_{t=1}^{T_{\max}} \mathcal{B}_{t,S,M} \tag{10}$$

To avoid favoring short sequences, the best sequence in $\mathcal{B}^{(\text{final})}$ is selected by the length-normalized score $\tilde{r}_{S,M}(\mathbf{b})$, given by

$$\tilde{r}_{S,M}(\mathbf{b}) = \frac{r_{S,M}(\mathbf{b})}{|\mathbf{b}|} \tag{11}$$

where $|\mathbf{b}|$ is the length of the sequence.

A.2 Experiment Details

Datasets. The models are trained on the WMT'14 English–German (En–De) training set (Bojar et al., 2014), which contains around 4M sentence pairs. We evaluate lexical control on three test sets, each constructed by filtering with terminology that is then used as the required constraints at inference: (1) the WMT'14 En–De test set filtered using Wiktionary terms, yielding 454 sentence pairs; (2) the WMT'17 En–De test set (Bojar et al., 2017) filtered using Wiktionary terms, yielding 727 sentence pairs; and (3) the same WMT'17 En–De test

set filtered using Interactive Terminology for Europe (IATE) terms (European Union, 2007), yielding 414 sentence pairs. In all cases, the exact terms used for filtering are provided to the model at test time as lexical constraints and must appear in the generated translations.

For the additional Zh–En experiment, we train models on the WMT'17 Zh–En training set (Bojar et al., 2017), which contains approximately 22M sentence pairs. Following prior lexically constrained MT work (Hokamp and Liu, 2017; Susanto et al., 2020), we construct the lexical-control test set from the English reference side by matching candidate constraints from public terminology and dictionary resources, including IATE and Wiktionary/Wiktextract. The final Zh–En lexical-control test set contains 200 sentence pairs. Since the goal is to evaluate decoder-side lexical control, we match constraints directly against English references and do not require automatic bilingual term alignment. Matching is performed in de-BPE word space, while the final constraint span is copied from the original BPE reference tokens. We keep unique, longest non-overlapping matches and filter frequent function-word phrases.

Model Configurations. For fair comparisons, we train our models and the baselines with the same settings. Specifically, we use the Transformer-base configuration as the backbone of the translation models; it has 6 encoding layers, 6 decoding layers, 8 attention heads, and a hidden size of 512. We use the same BPE vocabulary of size 40K for all models. To train the models, we follow the standard scripts in (Susanto et al., 2020), and use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 5e-4 and a batch size of 64K tokens. We do not conduct an additional hyperparameter search.

A.3 Reporting Protocol

Unless otherwise stated, MT results are reported from a single trained model/checkpoint for each system, following the standard setup in prior lexically constrained MT work (Hokamp and Liu, 2017; Susanto et al., 2020; Lee et al., 2021; Zeng et al., 2022). Since this MT setting uses standard training data, architectures, and evaluation protocols and typically has low variance, we report corpus-level metrics aggregated over each test set and averages for the decoding-space analyses.

A.4 Use of AI Assistants

AI assistants were used for grammatical edits, coding assistance, and revision planning. No original scientific content was generated by AI. The authors validated all AI-assisted output and retained full control over the method, experiments, benchmark curation, results, analyses, and final writing decisions.