Grid Beam Search for Constrained GPT-2 Decoding

Ilana Shapiro
Department of Computer Science
UC San Diego
La Jolla, CA 92092

Shubham Saha
Department of Mathematics
UC San Diego
La Jolla, CA 92092

Diya Lakhani
Department of Mathematics
UC San Diego
La Jolla, CA 92092

Shree Venkatesh
Department of Computer Science
UC San Diego
La Jolla, CA 92092

Runqiu Xu
Department of Mathematics
UC San Diego
La Jolla, CA 92092

Abstract

Constrained decoding is key to guiding neural language models toward structured outputs. Standard beam search lacks explicit constraint enforcement, limiting its use in structured tasks. We explore Grid Beam Search (GBS), a method ensuring predefined constraints in generated sequences. GBS was originally adapted for Neural Translation Models for use in translation tasks. In this work, we contribute an adaption of GBS for transformer architectures; specifically, GPT-2, and we evaluate its effectiveness in generative tasks. Furthermore, we fine-tune a pre-trained GPT-2 model on a corpus of Chekhov's stories and then compare model performances when subjected to lexical constraints. Our results demonstrated that we successfully integrated GBS for transformers, and we present a pipeline for generating text with constraints, given a prompt. Our subjective analysis also showed that GBS + fine-tuned GPT2 gave more interesting and meaningful domain-specific results than GBS + GPT2 alone. Our code can be accessed here.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

1 Introduction

Sequence models struggle to produce output that strictly adhere to predefined constraints. Constrained decoding seeks to modify the decoding process of sequence models by limiting valid outputs to a set of rules, which is important for structured NLP tasks. Such constrained decoding paradigms in prior research have included lexical rules Hokamp and Liu [2017]; Chen *et al.* [2022], predicate logic Lu *et al.* [2021], finite state automata Koo *et al.* [2024], and more recently, context-free grammars Geng *et al.* [2023]; Park *et al.* [2025, 2024] as constraint formalisms. All of these methods modify the model's *beam search* process. Beam search is one of the most popular algorithms used for text generation. At each step of generation, it maintains a set of n beams, where each beam represents one of the top n most probable sequences, or *hypotheses*, generated so far. The beams are expanded by considering all possible next tokens, and only the top n hypotheses are retained for the next step. Constrained decoding modifies these hypotheses that are added to the beams, ensuring only valid hypotheses are added at each step of token generation.

In this work, we explore Grid Beam Search (GBS) Hokamp and Liu [2017], which modifies beam search to enforce lexical constraints, or a list of strings that must appear in the output. We adapt GBS from its original implementation (Neural Machine Translation with RNNs) to GPT-2 to enable constraint-aware text generation from a prompt.

To judge model performance, a pre-trained GPT-2 model has been fine-tuned on a corpus of Chekov's stories scraped from Gutenberg. This fine-tuned model shall be used to in the constrained generation of texts based on a given prompt and a list of constraints. The generation of constraints is being automated by a random selection of short sentences from Chekhov's other compositions, not included in the training set, that are being passed through a translator which has been fined-tuned on the "Tale of Two Cities". This is to rephrase generated constraints so that they inherit the same meaning but don't necessarily fall inside the training corpus, making it a greater challenge for the model. The cumulation of our results demonstrate GBS's utility for structured text generation with LLMs. Our code can be accessed here.

2 Related Work

Recent advancements in constrained text generation have focused on different high-level approaches, including lexical constraints, grammatical constraints, and logical/formal constraints. These approaches aim to enhance the controllability of neural text generation while maintaining fluency and coherence.

One major direction is lexically constrained decoding, where specific words or phrases must appear in the generated output. A foundational work in this area is Grid Beam Search (GBS), introduced by Hokamp and Liu [2017],, which modifies traditional beam search to ensure predefined lexical constraints are respected during sequence generation. This method has influenced several later studies, including relation-constrained decoding Chen *et al.* [2022], which enforces relational dependencies between words to improve factual consistency in generated text. Unlike traditional decoding techniques that generate text in an unconstrained manner, RCD ensures that specific entities and their relationships appear in the output, making it particularly useful for tasks such as knowledge-grounded text generation and factual text completion.

Another line of research focuses on grammar-constrained decoding, which ensures syntactic correctness and structured outputs. Geng *et al.* [2023] proposed an approach that integrates grammatical constraints into NLP tasks without requiring additional fine-tuning. Instead of modifying a pretrained model, the method constrains the output at the decoding stage to align with predefined grammatical rules, making it efficient and adaptable to various applications. Park *et al.* [2025, 2024] further explored grammar-aligned decoding and efficiency improvements, making these constraints more computationally feasible for large-scale models. The approach focuses on directly aligning the decoding process with formal grammar rules, enhancing both syntactic correctness and fluency in generated text. Later on the paper improves the flexibility and computational efficiency of grammar-based decoding, addressing scalability challenges in constrained text generation.

Beyond lexical and grammatical constraints, formal logic-based approaches have been studied to impose higher-order structural constraints on generated text. Lu *et al.* [2021] introduced Neurologic Decoding, which uses predicate logic to guide text generation, ensuring logical consistency in outputs.

Unlike traditional decoding methods that generate text freely, Neurologic Decoding incorporates logical constraints during generation to ensure coherence, factual consistency, and adherence to structured reasoning.

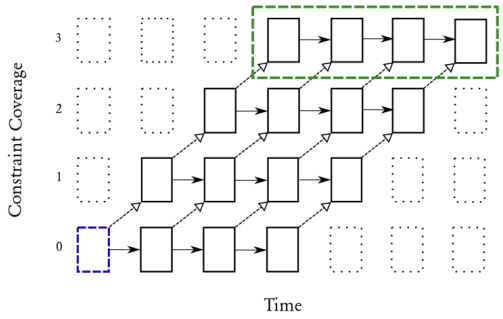
Similarly, Koo *et al.* [2024] investigated an automata-based constrained decoding framework that leverages finite-state machines (FSMs) and automata theory to impose structural constraints during language model decoding. The approach ensures that generated text adheres to predefined constraints while maintaining fluency and coherence.

Our work builds upon these efforts by integrating Grid Beam Search with GPT-2. The first component of the project dealt with data scraping and fine-tuning. The hyperparameters used to fine-tune a pretrained GPT-2 model were based on authors' prior work in Lakhani *et al.*. The data source for fine-tuning, constraint, and prompt generation being Gutenberg.

Along with the above, the method of score-boosting to "nudge" the model to contain constraints was inspired by Strobelt *et al.* [2021] and Keskar *et al.* [2019]. This was implemented to demonstrate how conventional Transformers fail to capture provided constraints despite being "nudged" to include the same.

Finally, in Grid Beam Search (GBS), the emphasis of this report, individual constraints may be single tokens or multi-word phrases, and any number of constraints may be specified simultaneously. More formally, given that the model is generating a sequence $S = \{y_0 \dots y_t\}$ these lexical constraints form the set $\{c_0 \dots c_n\}$, where each c_i is a sub-sequence $\{c_{i_0} \dots c_{i_j}\}$, that must appear somewhere in S. GBS builds on the traditional Beam Search algorithm for sequence generation. GBS seeks to organize decoding in such a way that we can constrain the search space to outputs which contain one or more of the pre-specified subsequences. The algorithm uses model's distribution both to "place" the lexical constraints correctly, and to generate the parts of the output which are not covered by the constraints.

In more detail, GBS extends the traditional time dimension of beam search (which tracks how long our hypotheses are) with a *constraint coverage* dimension (which tracks how many constraints have been satisfied). Thus, GBS forms a grid with constraint coverage on one axis, and time on the other (Figure 2). Each cell in the grid contains one beam of size n, each of which contains the top n hypotheses at timestep t and constraint coverage c.



Originally, GBS focused on seq2seq tasks (machine translation) with Neural Translation Machines (NTMs), and the constraints were user-provided inputs to correct the translations. At the time, NTMs were RNN/LSTM-type models, as the transformer was not yet invented. As such, to our knowledge, GBS has never been applied to a generative transformer-type model such as an LLM.

3 Methods

3.1 Baseline

To implement GBS, we base our approach on the code from the original paper Hokamp and Liu [2017], which we heavily modify and adapt to our model. We built a new pipeline enabling us to run any pretrained autoregressive language model with GBS. Given a prompt and a set of constraints (i.e. strings that must appear in the output), our pipeline forces the output of the model, when given the prompt, to include the given constraints. Our chosen baseline model is a pre-trained GPT-2 model

from HuggingFace (specifically, GPT2LMHeadModel), and we overwrite its built-in generation function with Grid Beam Search. Importantly, though, any other autoregressive model could have been easily swapped.

GBS was originally adapted for Neural Translation Machines (NTMs), which are stateful RNNs and thus have a profoundly different architecture than a transformer like GPT2. GBS exposes an interface for the model it is applied to with the functions: generate unconstrained, generate constrained (i.e. start a new constraint), and continue constrained (i.e. finish the current constraint). Most of our work focused on implementing this interface for GPT2. Throughout all functions, we maintain a constraint coverage vector that gets updated as we start and finish each constraint, to ensure that constraints don't get repeated in the output.

As we fill in the grid at each step of the generation, just like in original GBS, we select the best n hypotheses for the beam at each t, c entry of the grid for timestep t and constraint coverage c. In the generate unconstrained function, we feed in the input from the parent entry in the grid that we visited previously (i.e. from the parent hypothesis) into GPT2, and store the resulting log probabilities. We then use PyTorch's `top_k` function to select the n best tokens from this distribution, and save their scores. These n best tokens are appended to the previous input value to create n new hypotheses for the current beam, ranked by their log probability scores.

In the generate constrained function, we start a new constraint by iterating through the currently available constraints (i.e. those that have not already been covered in the previous hypothesis) and selecting the first token of that constraint as our next token. We append each token to the previous hypothesis to create the new set of hypotheses for our current beam, and get their log probability scores by indexing each token into the distribution returned by GPT2 from feeding in the previous hypothesis.

Finally, in the continue constrained function, we continue the previously unfinished constraint (this only gets called if we're in the middle of an unfinished constraint). We thus only create one hypothesis for this beam (the previous hypothesis plus the current token in the constraint we need to finish), and its score is determined exactly as in the generate constrained function.

At the end of GBS, the top row of the grid (i.e. when all constraints have been covered) is selected, and the highest scoring hypothesis is returned as the result. In the original GBS code, they stopped filling in the grid after all constraints were covered, even if the user specified a longer output. We modified GBS so the grid keeps getting filled in even after all constraints are covered, just by calling generate unconstrained, so we achieve sufficiently long and expressive outputs. We also wrote new test cases applicable to text generation (given a prompt and a set of constraints), as opposed to pure translation tasks as the original GBS code and paper did.

3.2 Finetuning

After completing the baseline with GPT-2 fully-integrated with the GBS framework we transitioned to finetuning with different text corpus. The baseline performed quite well and answered the prompt with coherent sentences containing all the constraints. However, finetuning will allow the model to produce more domain or task-specific outputs. In our case, we aim to ask our model to write a short story, given the first line of the story.

For the training text corpus we webscraped the short stories and plays written by Anton Chekhov from Project Gutenberg. This was done to expose the model to different themes that Chekhov worked with to ensure that all generated texts do not follow the same overarching narrative. We also implemented a score-boosting framework to encourage our model to make "better choices." To achieve this we assign a score to each token generated. The score values are from -10 to 10. End-of-sentence tokens are heavily penalized with -10 since as we do not want the sentence to end early. If tokens are continuing constraints then we boost the score of that token by +10. If we are generating a random new token from the overall text corpus then we boost by +5. While score boosting helps the model prioritize including the constraints, there is a trade-off between the quality of the sentences produces and the number of constraints being incorporated.

As we want to prompt our model to generate short story with a writing style similar to that of Anton Chekhov, we also developed a technique to generate random, suitable constraints. The constraints are generated by sampling short sentences from the texts "War and Peace" by Leo Tolstoy and a "Tale of

Two Cities" by Charles Dickens. These texts were also sourced by Project Gutenberg. These texts were written in around the same time period as Chekhov, however, by different authors who focus on different themes. We selected constraints from different texts to challenge our model and to ensure that constraints are not simply sentences directly available in Chekhov's writing.

4 Results

We ran three versions of our pipeline for the same prompt and constraints, with a 500 token generation limit. First we do GPT2 + GBS, then we do Fine Tuning + Score Boosting GPT2 (without GBS), and finally we do Fine Tuning + Score Boosting GPT2 + GBS.

4.1 Baseline Results

Prompt: "He had not seen the hussars all that day, but had heard about them from an infantry officer." Constraints: ['The peril of an', 'I could not see', 'Crunchers attention was here']

Story produced (constraints are highlighted): Story produced (constraints are highlighted):

He had not seen the hussars all that day, but had heard about them from an infantry officer. "**I could not see** the hussars all that day, but had heard about them from an infantry officer." Crunchers attention was here.

"I could not see the hussars all that day, but had heard about them from an infantry officer."

"I could not see the hussars all that day, but had heard about them from an infantry officer."

"I could not see the hussars all that day, but had heard about them from an infantry officer."

"I could not see the hussars all that day, but had heard about them from an infantry officer."

Crunchers attention was here.

The peril of an

4.1.1 Baseline Results Description

GBS does an effective job of including imposed constraints in generated texts. The coherence of generated text, however, is grossly affected by the lack of fine-tuning. As we shall observe in §4.3, the coherence issue gets addressed significantly when we apply GBS on a fine-tuned GPT-2 model.

4.2 Finetuning + Score Boosting Results

Prompt: "He had not seen the hussars all that day, but had heard about them from an infantry officer." Constraints: ['The peril of an', 'I could not see', 'Crunchers attention was here']

Story produced (constraints are highlighted):

He had not seen the hussars all that day, but had heard about them from an infantry officer. He

had been to see them all the evening, and had never seen them

again. The old man had not gone off to the river, but had been going up to the

highroad where the highroad was to be reached. He had no means of getting to the

highroad, and had only made his way along the highroad in the direction of the

highroad. He had no right to go on walking, and had to

make up his mind to go on walking. The old man was lying on the ground, exhausted, exhausted, with his eyes

filled with tears. It was evident from his face that he was exhausted,

and that he had no means of moving. The old man had not been the first to fall in love. At the

same time, the others had also fallen in love. The old man could not be sure whether this was a case of jealousy,

or if it was something else. He was afraid he would be forgotten by

the rest of the town. **The peril of an old man** who is ill, in spite of his age,

and who is never happy, is not to be expected, but it was not

easy to get on without him. His companions and acquaintances

were afraid of him, and they tried to make him go away, to make him

happy, and even to get him to come back. He did not

understand them, and could only say I dont understand. He had been in a continual state of dissatisfaction, and was continually

going to the highroad, and now, having no means of getting to the highroad,

he could not walk. He felt exhausted, and was afraid of being left behind

in the town. He was ashamed to live in the town, and could not

endure the solitude of the highroad. It was time to go away. He had not a long way to go, and could only look at the sky, the

white mountains, and the highroad, and think, How beautiful it is! The old man had not a long way to go, and could only look at the sky,

the white mountains, and the highroad, and think, How beautiful it is! He had not the strength to walk, and could only look at the sky,

the white mountains, and the highroad. He could not look at the sky, the white mountains,

and the highroad, and think, How beautiful it is! He was tired, and he could not get on

without his companions and acquaintances, and he could not get on without them. The old man could not get on without his companions and acquaintances, and he

could not get on without them. He felt as though he were dreaming, and was not

4.2.1 Finetuning + Score Boosting Results Description

As demonstrated above, our fine-tuned and score-boosted model has been successful in producing a somewhat-coherent story. However, one drawback of the score-boosting technique is quite apparent in the output text. That is, the model fails to include all constraints. Of the three constraints provided, only one is incorporated into the output text.

4.3 Finetuning + GBS Results

Prompt: "He had not seen the hussars all that day, but had heard about them from an infantry officer." Constraints: ['The peril of an', 'I could not see', 'Crunchers attention was here']

Story produced (constraints are highlighted): He had not seen the hussars all that day, but had heard about them from an infantry officer. who had come to see him. He had not seen the hussars all that day, but had

heard about them from an infantry officer who had come to see him. He had not

seen the hussars all that day, but had heard about them from an infantry officer who

had come to see him. He had not seen the hussars all that day, but had

heard about them from an infantry officer who had come to see him. He had not

seen the hussars all that day, but had heard about them from an infantry officer who

had come to see him. **I could not see** the hussars all that day. **Crunchers attention was here.**

The peril of an

4.3.1 Finetuning + GBS Results Description

As can be observed, running GBS on a fine-tuned GPT-2 model addresses the tradeoff issue between coherence and including constraints. The performance isn't flawless but supersedes preceding text generation obtained on pre-trained GPT-2 + GBS or Finetuned + Score-Boosted models.

5 Discussion

Our results demonstrate that GBS guarantees the appearance of all predefined lexical constraints in the generated output of a pertained, prompted GPT2 model. Our plug-and-play approach also sets the stage for applying GBS to *any* autoregressive text generation model, a significant step towards more diverse applications compared to the original GBS approach which purely looked at RNNs and machine translation. We also found that the *unconstrained* part of generated outputs is improved for a chosen domain/style by fine-tuning GPT2 to Chekhov's short stories, which increases the overall coherence of the output.

The fine-tuned model along with score-boosting is incapable of capturing all the constraints if the constraint is vastly unrelated to the prompt. GBS, however, is designed to capture all the constraints and maintain overall coherence by design. There are several choices of hyperparameters involved in fine-tuning as well as the choice of constraint length and number.

The chosen hyperparameters were set with prior optimal performance and fine-tuned in mind. A detailed description of fine-tuning hyperparameters can be found in Lakhani *et al.*.

There are several sophisticated methods for constrained decoding, other than score boosting and GBS. The literature on constrained decoding is vast and continues to be an active area of investigation as detailed in §2.

6 Conclusion

In conclusion, we have adapted Grid Beam Search from its original application in Neural Machine Translation with RNNs, to generative tasks from a prompt with transformer architecture models. We also demonstrated how score boosting and fine tuning, when combined with GBS, can enhance the results. In the future, it will be interesting to see how GBS can be combined with more complex constrained decoding paradigms for increasingly expressive constraint formalisms.

References

Xiang Chen, Zhixian Yang, and Xiaojun Wan. Relation-constrained decoding for text generation. *Advances in Neural Information Processing Systems*, 35:26804–26819, 2022.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore, December 2023. Association for Computational Linguistics.

Project Gutenberg. The Project Gutenberg ebook: Compilation of Short Stories by Chekhov.

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

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019.

Terry Koo, Frederick Liu, and Luheng He. Automata-based constraints for language model decoding, 2024.

Diya Lakhani, Ilana Shapiro, Shubham Saha, Shree Venkatesh, and Runqiu Xu. Custom fine-tuning and contrastive learning with bert.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Neurologic decoding: (un)supervised neural text generation with predicate logic constraints, 2021.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding, 2024.

Kanghee Park, Timothy Zhou, and Loris D'Antoni. Flexible and efficient grammar-constrained decoding, 2025.

Hendrik Strobelt, Jambay Kinley, Robert Krueger, Johanna Beyer, Hanspeter Pfister, and Alexander M Rush. Genni: Human-ai collaboration for data-backed text generation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):1106–1116, 2021.