Making Grid Beam Search Less Greedy

Anonymous ACL submission

Abstract

A common formalism for constraining the output of autoregressive text generation models involves *lexical constraints*, words or phrases which are required to occur in the generated text. DFA-constrained beam search and grid beam search are two widely used paradigms for decoding from autoregressive models while enforcing lexical constraints. As the former approach requires a number of forward passes exponential in the numberof constraint tokens, it is often dispreferred to the latter, which requires only linearly many forward calls. However, while grid beam search achieves an exponential speedup, it does so in a manner which does not treat all of the constraints equally. In this paper, we demonstrate that grid beam search is biased to incorporate easier-to-satisfy constraints first, leaving harder constraints to the end of the sequence. This contrasts with DFA-constrained beam search, which exhibits no such bias. To address this shortcoming, we propose fair grid beam search, a modification to grid beam search which avoids this bias while still requiring only linearly many forward passes. Experimentally, we confirm grid beam search's bias on two constrained generation tasks, finding significant differences in how it orders constraint tokens as compared to DFA-constrained beam search and fair grid beam search. Furthermore, we find that fair grid beam search not only fixes grid beam search's bias, but finds higher-probability strings in the process.

1 Introduction

Given the tremendous success large language models (LLMs) have had in tasks across natural language processing (NLP), a common target of NLP research is in understanding how to best use LLMs for specific tasks. As language models define distributions over strings of natural language, while most tasks in NLP do not involve modeling arbitrary natural language, techniques for applying LLMs tend to involve conditioning the LLM distribution such that the resulting conditional distribution over strings can be interpreted as a distribution over task-specific predictions. For classification tasks like sentiment classification (e.g., Zhang et al., 2024b), this might simply involve conditioning the LLM to produce the name of a valid label, while for structured prediction tasks like parsing (Drozdov et al., 2022), more sophisticated approaches may be required in order to interpret the LLM's distribution over strings as a distribution over valid structures.

The most common technique in this direction is prompting, where LLM output is conditioned on a *prompt*, which usually takes the form of natural-language instructions. However, as prompting involves conditioning by natural-language instructions, its success relies on the natural-language understanding (NLU) capabilities of the autoregressive model. This is often not a problem when a) the model exhibits strong NLU capabilities, b) the instructions are easy-to understand and easy-to-follow, and c) some probability of failure is acceptable, but, when such assumptions do not hold, other approaches to conditioning may be taken.

One such alternative is *constrained decoding* (Anderson et al., 2017; Hokamp and Liu, 2017; Post and Vilar, 2018; Lu et al., 2021). In this paradigm, a formal language of valid strings is defined, and the language model is conditioned on the event that the generated string is an element of this language. One particularly well-explored setting for constrained decoding involves *lexical constraints*: constraint languages that stipulate strings must contain a particular substring, and intersections of such languages. In other words, a lexical constraint stipulates that a particular word or phrase must occur in the model output, and multiple such lexical constraints can be enforced simultaneously.

Grid beam search (Hokamp and Liu, 2017) and DFA-constrained beam search (Anderson et al., 2017) are two widely used methods for decoding

from autoregressive models while enforcing lexical constraints. Both modifications to standard beam search (Lowerre, 1976), these algorithms ensure that decoding hypotheses make progress towards fulfilling all constraints by maintaining multiple beams of hypotheses. We will compare these algorithms in detail in Section 2; for now, it suffices to note that, for k lexical constraints of bounded length, grid beam search requires $O(k)$ autoregressive forward passes per token, while DFA-constrained beam search requires $O(2^k)$ forward passes. As such, grid beam search enjoys significantly more popularity.

In this paper, we draw attention to an underappreciated problem with grid beam search: It is unnecessarily *greedy*, and prefers to satisfy easy lexical constraints, i.e. common constraint words, first, leaving harder constraints until the end. Hence, common constraint words will occur *earlier* in generated sequences than rare ones. In addition to illustrating this problem, we also present a solution: fair grid beam search, a modification to grid beam search which maintains the original's efficiency while eliminating the bias. This leads to higher-probability strings in the end. In order for others to easily apply our decoding method, we release our code as an open-source python library.[1]

In Section 2, we discuss existing approaches to constrained decoding, with a heavy focus on DFA-constrained beam search and fair grid beam search, which are particularly relevant as background to our current work. Section 3 provides conceptual argumentation for the greedy bias of grid beam search. Section 4 presents fair grid beam search as an alternative to grid beam search that avoids this bias. In Section 5, we present two experiments which empirically compare DFA-constrained beam search, grid beam search, and fair grid beam search, empirically validating our claims about grid beam search's ordering bias and about fair grid beam search's improvements. Finally, we conclude the paper in Section 6.

2 Constrained Decoding

We now discuss previous literature on constrained decoding in autoregressive models. We focus on Anderson et al. (2017) and Hokamp and Liu (2017), as we directly build on these prior works.

[1]Anonymized code included in the supplementary materials for review, with a finalized library forthcoming.

2.1 DFA-Constrained Beam Search

We paraphrase DFA-constrained beam search as it was introduced in Anderson et al. (2017). While the approach was simply termed "constrained beam search" in this original publication, we refer to it as DFA-constrained beam search for the sake of disambiguity.

We begin with the observation that languages induced by sets of lexical constraints are regular. Let Σ be our alphabet of tokens, and let $\boldsymbol{c} = \langle c^1, c^2, \cdots, c^l \rangle \in \Sigma^*$ be a lexical constraint – a token sequence which must occur in all generated strings. The regular expression $\Sigma^* c^1 c^2 \cdots c^l \Sigma^*$ describes exactly those strings which satisfy the constraint $\boldsymbol{c}$. As regular languages are closed under intersection, combinations of lexical constraints are similarly regular: For any finite set of k lexical constraints $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_k\}$, we can define a regular language

$$L = \bigcap_i \left(\Sigma^* c_i^1 c_i^2 \cdots c_i^l \Sigma^* \right)$$

where a string $\boldsymbol{s} \in L$ iff $\boldsymbol{s}$ satisfies all lexical constraints.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a minimal DFA for L. Without an explicit construction, it can be seen that, in the worst case, M requires a number of states exponential in the number of constraints, as the automaton must "remember" at every time step which subset of constraints has already been satisfied.

Given such a minimal automaton M, DFA-constrained beam search associates with each DFA state $q \in Q$ a beam of hypotheses $B^t(q) \subseteq \Sigma^t$, indexed by time step t. Conceptually, each hypothesis is a token sequence which could potentially form a prefix to the final decoded string. The beams are initialized to $B^0(q) = \{\boldsymbol{\varepsilon}\}$ for the initial state $q = q_0$, and $B(q) = \varnothing$ for all other $q \neq q_0$.

During each decoding step t, new hypotheses $\boldsymbol{h} \in \Sigma^t$ are built by considering every possible continuation token α for each existing hypothesis $\boldsymbol{h}' \in \Sigma^{t-1}$ across all beams, forming a hypothesis set

$$H^t = \{\boldsymbol{h}'\alpha \mid \boldsymbol{h}' \in \bigcup_q B^{t-1}(q); \alpha \in \Sigma\}.$$

Each such $\boldsymbol{h} \in H^t$ can be assigned to a DFA state $\delta^*(q_0, \boldsymbol{h}) \in Q$, where δ^* is M's extended transition function. From this, we can define state-wise hypothesis sets

$$H^t(q) = \{\boldsymbol{h} \mid \boldsymbol{h} \in H^t, \delta^*(q_0, \boldsymbol{h}) = q\}$$

of all hypotheses $\boldsymbol{h}$ which *could* be elements of the beam $B^t(q)$. To obtain the actual beam, we simply retain the top β hypotheses in that state-wise set according to our autoregressive model's probability function $P(\boldsymbol{h})$:

$$B^t(q) = \underset{\boldsymbol{h} \in H^t(q)}{\arg\,\text{top-}\beta}\, P(\boldsymbol{h})$$

Since we would only like to generate strings in L, we only consider candidate strings originating from accepting states' beams as model output. As with standard beam search, we proceed until our best candidate string is more probable than our highest-probability hypothesis.

2.2 Grid Beam Search

Grid beam search, as it was first presented in Hokamp and Liu (2017), was described in terms of an interface for building new hypotheses with functions `generate`, `start`, and `continue`. We reanalyze Hokamp and Liu's (2017) algorithm in terms of finite-state automata: in particular, the algorithm describes a) a method for constructing a non-deterministic finite-state automaton (NFA), and b) a decoding procedure, analogous to DFA-constrained beam search, for generating strings accepted by this NFA. Importantly, this decoding method, when applied to the NFA obtained from a), leads to a time complexity linear in the number of constraint tokens, as contrasted with DFA-constrained beam search's exponential runtime.

In this work, we will consider b) to be the central "essence" of grid beam search, and treat a) to be an implementation detail. In fact, we note that grid beam search's decoding scheme is equally applicable to the minimal DFA for the constraint language L, with identically linear time complexity. In light of this, we will only discuss grid beam search's decoding procedure here, and, for the remainder of this paper, we will consider grid beam search to operate on a minimal DFA for L. In Appendix 6, we discuss the specific NFA constructed by Hokamp and Liu (2017), and discuss how to map our construction to nondeterministic automata.

We start with the DFA M. We assign to each state q a *depth* $d(q) \in \mathbb{N} \cup \{\infty\}$ equal to the minimum number of transitions needed to reach an accepting state starting from that state (with accepting states being assigned a depth of zero, and coinaccessible states being assigned infinite depth). Note that this notion of depth is equivalent to the notion of constraint coverage presented in Hokamp and Liu (2017), in that both measure the minimal number of tokens which must be generated before all constraints will be satisfied.

Decoding can then be defined similarly to with DFA-constrained beam search, with one major difference: instead of maintaining one beam of hypotheses $B(q)$ for each *state*, we instead maintain one beam of hypotheses $B(d)$ for each *distinct depth value*. The beams are initialized to $B^0(d) = \{\boldsymbol{\varepsilon}\}$ for the depth of initial state $d = d(q_0)$, and $B(d) = \varnothing$ for all depth values $d \neq d(q_0)$. Hypothesis sets are defined equivalently as

$$H^t = \{\boldsymbol{h}'\alpha \mid \boldsymbol{h}' \in \bigcup_d B^{t-1}(d); \alpha \in \Sigma\}.$$

However, instead of defining state-wise hypothesis sets, we define depth-wise hypothesis sets:

$$H^t(d) = \{\boldsymbol{h} \mid \boldsymbol{h} \in H^t, d(\delta^*(q_0, \boldsymbol{h})) = d\}$$

Forming beams proceeds identically by selecting the β best hypotheses from each of these hypothesis sets:

$$B^t(d) = \underset{\boldsymbol{h} \in H^t(d)}{\arg\,\text{top-}\beta}\, P(\boldsymbol{h}).$$

While DFA-constrained beam search required we pick our candidate strings from accepting states' beams, in grid beam search, we select candidates from the beam for depth zero.

This construction maps the exponentially-many states of M to a linearly-many equivalence classes over states, and only maintains one beam for each equivalence class. By specifying these equivalence classes in terms of depth, it is ensured that each partial candidate from each beam will have at least one child in a beam of lower depth, inductively ensuring that some candidates satisfying all constraints will be found.

2.3 Other work on constrained decoding

In addition to the two works presented above, a number of other approaches have been proposed for modifying beam search for constrained decoding. Two particularly prominent examples of this are Post and Vilar (2018), who modify grid beam search by varying beam sizes dynamically, and Lu et al. (2021), who present a beam-search-based algorithm for decoding under unions, intersections, and negations of lexical constraints. Apart from beam search, other approaches for constrained decoding from autoregressive models include transformations of the task into optimization in a continuous space (Kumar et al., 2021; Dathathri et al.)

and text generation via Metropolis-Hastings sampling (Miao et al., 2019).

3 The Greedy Bias of Grid Beam Search

We claim that grid beam search suffers a "greedy" bias in that it prefers hypotheses which incorporate frequent constraint tokens before infrequent ones, at the expense of average hypothesis log-likelihood.[2] In this section we present a conceptual argument as to why this occurs. We will make this argument in terms of competition between hypotheses which occurs during beam search, wherein the retention of a given hypothesis from one step to the next depends on the set of alternative hypotheses present in the same beam.

Suppose for simplicity that all constraints are single-token lexical constraints. In DFA-constrained beam search, since a beam is maintained for each DFA state, there is only competition between hypotheses which have have satisfied exactly the same set of constraints. For grid beam search, by contrast, there is competition between hypotheses which have completed the same *number* of constraints, but different hypotheses may have completed different sets of constraints.

Figure 1 illustrates how this competition can lead to decoding bias. As words' frequencies vary considerably in natural language (Zipf, 1949; Clauset et al., 2007), some of our constraint words will surely be more common than others. While specific contexts will contribute a large amount of variance, these word frequencies should affect the average autoregressive probabilities assigned to our constraint words, with common constraint words receiving higher mean autoregressive probabilities than rare ones. On average, we would expect hypotheses including rare constraint tokens to be assigned a lower model probability than hypotheses including only common constraint tokens.

Under grid beam search, we maintain a beam for each depth, and within each beam there is competition between hypotheses which have completed the same number of constraint tokens. Thus, in the presence of constraint tokens of varying frequency, for early (high-depth) beams, there will be direct competition between hypotheses containing only common constraint tokens, and those containing rare constraint words. The hypotheses containing rare constraint words will tend to lose this competition, and these early beams will preferentially fill with hypotheses satisfying only easy constraints, leaving rare constraint words for later beams.

Another perspective on this problem is that, when comparing hypotheses in grid beam search, we only account for the likelihood of the tokens that have already been generated, and not the likelihood of tokens that are yet to come. While this is true of beam search in general,[3] in constrained decoding, we have a priori knowledge about what tokens are yet to come, and we can take advantage of this knowledge by rewarding hypotheses which will likely have an easier time completing the remainder of their constraints. From this perspective, the tendency of grid beam search to complete easy constraints first is merely a symptom of an underlying inefficiency in finding high-likelihood candidates.

In section 5, we will demonstrate both of these observations empirically, namely, grid beam search *does* prefer to satisfy easy constraints first, and this *does* lead it to finding strings with lower average log-likelihood. But first, we will propose a simple fix to make grid beam search less greedy.

4 Methods: Fair Grid Beam Search

In this section, we present a variant of grid beam search which addresses the deficiencies discussed above. As our variant shows no unfair preference for incorporating high-frequency constraints earlier, we term it *fair grid beam search*.

4.1 Construction

Our construction is largely similar to that for grid beam search, with one key difference – we associate with each state q of M a cost $\mathcal{C}(q)$, representing the expected difficulty of reaching an accepting state from q, and account for this cost when comparing hypotheses. We accomplish this by weighting all arcs of M with $-\ln P(\alpha)$, the negative log unigram likelihood of that arc's symbol (token) α. We then define the cost $\mathcal{C}(q)$ of each state q to be the minimum distance to an accepting state in the underlying directed graph. By this construction, accepting states have a cost of zero, and non-co-accessible states have infinite cost. For notational convenience, we can also define the cost of a string

[2] This claim is not entirely novel to the present work – Lu et al. (2021) allude to this bias "towards sequences satisfying constraints greedily" in their discussion of grid beam search. However, to our knowledge, no other works have investigated this bias in depth, nor presented a direct modification to grid beam search to account for it.

[3] In fact, this can be seen as the central simplifying assumption that separates left-to-right approximate decoding schemes from exact decoding.

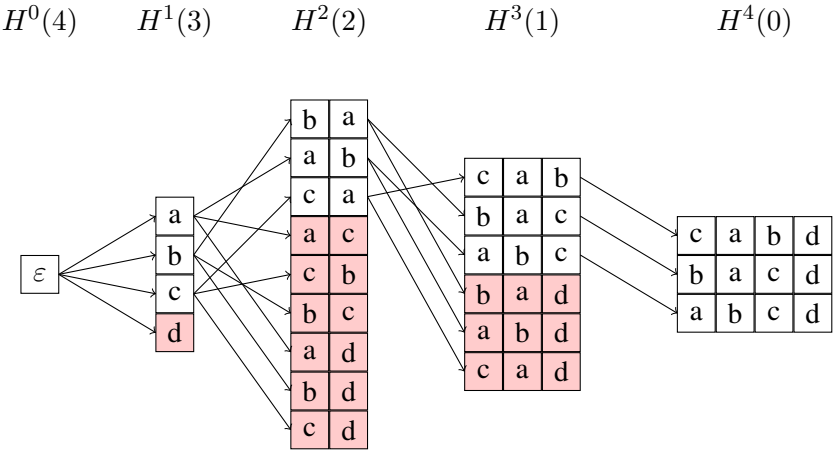

Figure 1: An illustration of the greedy bias in grid beam search with beam width 3. Consider a setting with four constraint tokens: a, b, c, and d, with unigram frequencies ordered $p(\text{a}) > p(\text{b}) > p(\text{c}) > p(\text{d})$, and with strings starting with low-frequency tokens being slightly more probable than those starting with high-frequency tokens. Assuming no non-constraint tokens, our highest-probability string should be "dcba." As grid beam search discards early hypotheses with low-frequency constraint tokens, it fails to find this global optimum, and prefers hypotheses which save the hardest constraint 'd' for last. For clarity, we only illustrate the generation of constraint tokens, not non-constraint tokens. This results in a depiction of a "diagonal slice" of the grid of beams, where time step and beam depth vary together.

$\boldsymbol{s}$ as $\mathcal{C}(\boldsymbol{s}) = \mathcal{C}(\delta^*(q_0, \boldsymbol{s}))$, the cost associated with the state reached while processing $\boldsymbol{s}$ through M.

Formally, fair grid beam search only differs from grid beam search in the construction of beams: Rather than comparing hypotheses $\boldsymbol{h}$ by probability $P(\boldsymbol{h})$, we instead compare the quantity $\ln p(\boldsymbol{h}) - \mathcal{C}(\boldsymbol{h})$:

$$B^t(d) = \underset{\boldsymbol{h} \in H^t(d)}{\arg\,\text{top-}\beta} \ln P(\boldsymbol{h}) - \mathcal{C}(\boldsymbol{h}).$$

As with grid-beam search, we will still be making "unfair" comparisons between hypotheses which have completed different subsets of constraints. However, this cost term compensates for this by acting as a heuristic for the difficulty of completing the remainder of the constraints. Hypotheses which have incorporated infrequent constraint tokens will have lower cost than hypotheses which have only incorporated frequent ones.

Our constructions requires access to unigram probabilities from our autoregressive model's distribution $P(t)$. Importantly, these unigram probabilities are conditioned on the prompt being used as well as the constraint setting being used, meaning we can't rely on precomputed corpus statistics for these unigram distributions. Luckily, these unigram probabilities can easily be obtained in practice by simply averaging all next-token distributions seen so far during beam search. When many texts are to be generated using a similar prompt and constraint setting, this means that the first text must be generated with standard grid beam search, but each subsequent text will be generated with unigram probabilities obtained while generating all previous texts. Of course, the first text may then be re-generated using the unigram probabilities obtained through the entire generation process.

4.2 Time Complexity

Assume we are interested in generating sequences of a maximal length n, decoding with k lexical constraints, each of l tokens, and with a beam width β. Treating our autoregressive model as a constant-time oracle, grid beam search has a time complexity linear in all of these quantities, i.e. $O(nl\beta k)$. This can be seen by noting that, for each of n time steps, we maintain $l \times k$ beams, each containing β hypotheses, and that we carry out one autoregressive call for each hypothesis that is part of a beam. Conversely, DFA-constrained beam search has a time complexity exponential in the number of constraint tokens, $O(nl\beta 2^k)$: At each time step, we maintain one beam of β hypotheses for each DFA state, but in the worst case we may have $l \times 2^k$ states – each state must remember which of the 2^k subsets of constraints has already been satisfied, and must remember how many of the l tokens of the currently-in-progress constraint have been

completed.

If the state costs $\mathcal{C}$ are precomputed, fair grid beam search's time complexity is identical to that of grid beam search: the only appreciable difference is the need to obtain hypothesis costs each time step, and this can be done in constant time if the state associated with each hypothesis is cached (constant-time determination of hypothesis state from the parent's state, and constant-time lookup of the cost from the state). However, the precomputation of the costs $\mathcal{C}$ involves solving a single-source shortest path problem on a graph of $|V| = l \times 2^k$ vertices. Using Dijkstra's algorithm with a Fibonacci heap (Fredman and Tarjan, 1987), this can be done in $O(|V| \log |V|)$, giving the entire algorithm a time complexity of

$$O(|V| \log |V| + nl\beta k)$$

$$= O\left(l2^k (k + \log l) + nl\beta k\right).$$

This is, of course, ultimately exponential in the number of constraints, and so appears to offer no benefit over DFA-constrained beam search. However, only the precomputation of $\mathcal{C}$ is exponential-time, and this precomputation does not involve any calls to the autoregressive model. Thus, in typical use cases, where GPU-based model inference capacity is the primary limiting resource, this CPU-based precomputation step is unlikely to be a limiting factor to model throughput for relatively small numbers of constraints.

5 Experiments

In this section, we discuss two experiments we carry out which validate three claims we make:

a) that grid beam search, compared to DFA-constrained beam search, prefers to satisfy easy constraints first,

b) that fair grid beam search corrects this bias, and

c) that fair grid beam search finds higher-probability strings than grid beam search.

Our first experiment, performed with a large number or randomly-generated constraint settings, directly tests and validates all three claims, albeit in a somewhat artificial setting, while our second experiment, performed at a smaller scale with manually-curated constraint sets, compares grid beam search to fair grid beam search in a more naturalistic constrained decoding setting, and further validates our affirmative answers to claims b) and c).

5.1 LLM Generation with Many Random Constraints

Our first experiment is designed to highlight the tendency for grid beam search to incorporate common constraint tokens before rare ones, as contrasted with DFA-constrained beam search, which does not exhibit this bias, and fair grid beam search, which corrects for this bias. This experiment involves generating short texts from the TinyLlama-1.1B language model (Zhang et al., 2024a) with randomly chosen lexical constraints. For each generation task, five English words are selected uniformly randomly from the 5000 most frequent words in the Corpus of Contemporary American English (COCA) (Davies, 2008). In order to make the task setting conceptually simple and ease the interpretability of results, we limit ourselves to constraint words which map to a single model token, reselecting whenever we choose a multi-token word. The model is then prompted to "write a one-sentence story," with no information about the constraint words provided in the prompt. We select 1000 such five-word constraint sets, and, for each set, generate a sentence using the three constrained decoding methods. Inference is performed in four independent "runs" of 250 generation tasks each – this detail is of consequence for grid beam search, where unigram statistics are collected separately for each of these independent runs.

We are interested in the correlation between constraint word frequency and relative position within the generated sentence: we hypothesize that grid beam search should exhibit a negative correlation (constraint tokens with high frequency should have low average token index, and vice versa). In order to avoid sensitivity to sentence length, we formalize this in terms of a notion of *relative index* – within each generated sentence, the first constraint token to appear is assigned a relative index of 1, the second a relative index of 2, and so forth, up to 5 for the final constraint token to appear. Then, for each constrained decoding method, we can analyze the Spearman rank correlation ρ between constraint token frequency and relative index across all 1000 generated texts.

Table 1 shows that, as hypothesized, grid beam search exhibits a small, yet highly significant ($p < 10^{-11}$) negative correlation: on average, low-frequency constraint words come later than high-frequency constraint words in sequences. For DFA-constrained beam search and fair grid beam search,

Method	ρ	H
DFA-constrained	(0.0180)	58.40
Grid	-0.100	60.81
Fair grid	(0.0271)	60.48

Table 1: Results for our experiments with random constraint words. For DFA-constrained beam search, grid beam search, and fair grid beam search, we report Spearman correlation coefficients ρ between word frequency and relative index, and decoding entropy H. Parenthesized correlations are found to be statistically insignificant at $p < 0.05$, while the correlation for grid beam search is significant at $p < 10^{-11}$. A two-tailed paired t-test found significant pairwise differences ($p < 0.0001$) between all three methods' decoding entropies.

we are unable to find any significant correlation at a significance level of $p < 0.05$.

In addition to investigating correlations, we also compare the model-assigned probabilities $P(\boldsymbol{s})$ of the decoded strings $\boldsymbol{s}$. As all methods act as decoding layers for the same underlying autoregressive model, we can compare these probabilities to directly compare how well these methods do at constrained decoding – better decoding methods should find, on average, higher probability strings. We quantify this in terms of *decoding entropy* H over the set of all generated strings S:

$$H = -\frac{1}{|S|} \sum_{\boldsymbol{s} \in S} \ln P(\boldsymbol{s}).$$

In broad terms, methods which attain a lower decoding entropy do a better job at decoding with constraints.

Decoding entropies are also listed in Table 1. Across the three methods, DFA-constrained beam search achieves the lowest decoding entropy. This is to be expected, since DFA-constrained beam search maintains a much larger set of beams, and therefore hypotheses, than the other two methods. However, between grid beam search and fair grid beam search, fair grid beam search attains a lower decoding entropy, despite maintaining the same number of hypotheses. A two-tailed paired t-test found this difference to be statistically significant ($p < 0.0001$).

5.2 CommonGen

For our second experiment, we aim to test a more natural constrained decoding setting with a larger autoregressive model, Llama-3.1 8B Instruct (LLama Team, 2024). Instead of using a random set of constraint words for each generation task, we make use of CommonGen (Lin et al., 2020), a dataset designed to challenge lexically constrained generation models, which specifies 400 distinct constrained generation tasks, each with a set of semantically-related "concepts" as constraints. These concepts, formally specified as the combination of a word and a part of speech, roughly correspond to lexemes. Thus, for example, the constraint `run_V` could be satisfied by sentences containing the words "run," "ran," "runs," or "running."

To allow for such varied surface realizations, we use the LemmInflect Python library (Jascob, 2022) to obtain a set of inflected forms for each concept, and further expand these sets by including variations in capitalization. In contrast to our previous experiment, we allow for multi-token surface realizations. For each concept, we construct a regular expression for the the union of all surface realizations, and we take the intersection of these unions as our constraint language. We use a minimal DFA for this constraint language to guide our beam search decoders.

While the CommonGen task is typically framed in a setting where models are explicitly told the constraint words in a prompt, we instead choose a constraint-blind setting, where the constraints are only enforced by the decoding scheme, and not known to the language model itself. Although this setting precludes numerical comparisons to prior work on CommonGen, and in fact makes the task significantly harder, it allows us to better analyze the effects of the decoding method in isolation, without any interference by the NLU capabilities of the language model.[4]

The automata we obtain with this approach have significantly more states than those for our previous experiment. For this reason, we do not test DFA-constrained beam search in this setting, and only compare grid beam search to fair grid beam search.

As with our first experiment, we compare two values across decoding schemes: the Spearman rank correlation ρ between constraint word frequency and relative index, and decoding entropy H. As we allow varied surface realizations of constraint words, we take as word frequencies the sum of all surface realizations frequencies in COCA.

[4]Preliminary experiments showed that including constraint concepts in the prompt significantly affected the unigram probabilities of the tokens comprising those constraints, leading to erratic performance when using fair grid beam search with precomputed unigram probabilities.

Method	ρ	H
Grid beam search	-0.28	49.22
Fair grid beam search	-0.06	48.14

Table 2: Results from our experiments on the CommonGen dataset for grid beam search and fair grid beam search. As before, we report Spearman correlation coefficients ρ between word frequency and relative index, and decoding entropy H. A two-tailed paired t-test found the difference between the two methods' decoding entropies to be statistically significant at $p < 0.0001$. Both correlations were found to be statistically significant at $p < 0.05$, but they were also found to differ significantly from one another via bootstrapping with a two-tailed binomial test ($p < 0.00001$).

Table 2 lists our results from this experiment. In summary, we reconfirm the two hypotheses we sought to validate with this experiment: that grid beam search preferentially satisfies easy constraints before hard ones, and that fair grid beam search achieves a lower decoding entropy than fair grid beam search. Of note, in this setting, both grid beam search and fair grid beam search achieve a significant negative correlation coefficient, just of vastly different magnitudes. This is not particularly surprising, as we have no a priori reason to expect that there should be exactly zero correlation between frequency and sentence position in natural language. However, the significant ($p < 0.00001$) difference between the two correlation coefficients provides clear evidence that grid beam search introduces its own bias in this regard.

Surprisingly, the decoding methods' effect on both constraint ordering and decoding entropy appear to be much stronger in this experiment than they had been for our setting with a smaller language model and random constraints. Grid beam search has a correlation coefficient of -0.28 (compared to -0.10 for the previous experiment), and fair grid beam search improves upon grid beam search by over one nat in decoding entropy. We had expected these effects to be more subtle in this setting: The order of words is a consequence of both biases introduced by the decoding method as well as semantic effects enforced by the language model, and, in a setting with a more competent language model and stronger semantic relations between constraint words, we did not expect the decoding method's bias to affect the results so dramatically. While we suspect the amplified effects seen in this setting may result from the increased complexity of the constraining automaton (which includes multi-token constraints and constraints with multiple surface realizations), further experiments would be necessary to fully understand what circumstances modulate the strength of grid beam search's bias. Nonetheless, this experiment provides strong evidence that the problem we point out in grid beam search, and the solution we present with fair grid beam search, are of practical relevance in realistic constrained decoding settings.

6 Conclusion and Future Work

In this work, we discuss the greedy bias of grid beam search and how to fix it. We present a conceptual argument as to why grid beam search preferentially satisfies easy constraints first, and experimentally demonstrate that it in fact does so in practice, biasing the constraint orderings of generated sentences. To correct for this bias, we present fair grid beams search, a slight modification to the decoding algorithm that rewards hypotheses for including difficult constraints. Experimentally, we demonstrate that fair grid beam search not only fixes grid beam search's bias in constraint ordering, but that it finds higher-probability strings in the process. While the time complexity of fair grid beam search is worse than grid beam search, all extra computation is in a precomputation step that requires no access to the underlying autoregressive model, meaning that the improvements provided by fair grid beam search essentially come "for free" in settings where model throughput is the computational bottleneck.

A theme of this work which might lead to future improvements in constrained decoding is the development of a more nuanced notion of progress towards completion and comparability of hypotheses. In grid beam search, progress was measured purely in tokens, while our present work demonstrates the need to also account for the rarity of these tokens. Future work could take a more detailed view of both how to quantify completion progress, and under which circumstances hypotheses of varying completion progress can be compared. This might lead to more general approaches, where the number of beams can be adjusted freely, and hypotheses can be assigned to beams in a way that will maximize the fairness of competition between candidates.

Limitations

One major limitation of this work is our inability to explain the exaggerated strength of grid beam search's bias on the CommonGen experiment versus the more muted results in our experiment with random constraints. While we expect this might be a consequence of the more complicated automaton, incorporating multi-token constraints and varied surface forms, more work would need to be done to understand how this ordering bias is affected by automaton structure. Indeed, we also neglect automata with Kleene stars in this work, and so the behaviors of the different decoding algorithms are still unknown for many types of constraint languages.

An additional limitation to this work is the lack of clarity regarding interactions between fair grid beam search and direct mentions of constraint words in prompt. While preliminary experiments showed that fair grid beam search performed poorly when constraint words were mentioned explicitly in prompts, we were unable to quantify this interaction more precisely, nor were we able to propose an alternative that was stable in such circumstances.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945, Copenhagen, Denmark. Association for Computational Linguistics.

Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. 2007. Power-law distributions in empirical data. Cite arxiv:0706.1062Comment: 43 pages, 11 figures, 7 tables, 4 appendices; code available at http://www.santafe.edu/ aaronc/powerlaws/.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Mark Davies. 2008. The corpus of contemporary american english (coca). Online database. Available online at `https://www.english-corpora.org/coca/`.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2022. Compositional semantic parsing with large language models. *CoRR*, abs/2209.15003.

Michael L. Fredman and Robert Endre Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615.

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, Vancouver, Canada. Association for Computational Linguistics.

Brad Jascob. 2022. LemmInflect: A python module for english lemmatization and inflection. Version 0.2.3.

Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. 2021. Controlled text generation as continuous optimization with multiple constraints. *Advances in Neural Information Processing Systems*, 34:14542–14554.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

LLama Team. 2024. The lLama 3 herd of models. *Preprint*, arXiv:2407.21783.

Bruce T. Lowerre. 1976. *The Harpy speech recognition system*. Ph.D. thesis, Carnegie-Mellon University.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online. Association for Computational Linguistics.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

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, New Orleans, Louisiana. Association for Computational Linguistics.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024a. Tinyllama: An open-source small language model. *Preprint*, arXiv:2401.02385.

Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Pan, and Lidong Bing. 2024b. Sentiment analysis in the era

of large language models: A reality check. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3881–3906, Mexico City, Mexico. Association for Computational Linguistics.

George K. Zipf. 1949. *Human Behavior and the Principle of Least Effort*. Addison-Wesley.

A Grid beam search as a finite-state automaton

Hokamp and Liu (2017) define grid beam search in terms of an interface of three functions: `generate`, `start`, and `continue`, and in terms of hypotheses, which may either be *open* (not "working on" any constraint) or *closed* (currently "working on" one particular constraint. We reinterpret this description as describing the construction of a nondeterministic finite-state automaton. We take the states of our automaton to be complete answers to the following set of questions:

a) Which subset of constraints has already been completed?

b) Which constraint, if any, is currently being worked on?

c) If applicable, how many tokens of it have already been completed?

That is, each state q_i is a triple of answers to questions a), b), and c). For a constraint set C, the stating state, q_1, answers these questions as ($\varnothing$, none, not applicable), and the sole accepting state answers these questions (C, none, not applicable).

The three functions `generate`, `start`, and `continue` define the transitions of the automaton. For any possible subset of constraints $A \in 2^C$, and for any symbol $\alpha \in \Sigma$, `generate` defines the self loop transition

$$(A, \text{none}, \text{not applicable}) \xrightarrow{\alpha} (A, \text{none}, \text{not applicable}).$$

For a constraint $c_i \notin A$, `start` defines transitions of the form

$$(A, \text{none}, \text{not applicable}) \xrightarrow{c_i^1} (A \cup \{c_i\}, \text{none}, \text{not applicable})$$

when c_i is a single token constraint, and

$$(A, \text{none}, \text{not applicable}) \xrightarrow{c_i^1} (A, c_i, 1)$$

otherwise.

Finally, `continue` generates transitions of the form

$$(A, c_i, j-1) \xrightarrow{c_i^j} (A \cup \{c_i\}, \text{none}, \text{not applicable})$$

when $|c_i| = j$, and

$$(A, c_i, j-1) \xrightarrow{c_i^j} (A, c_i, j)$$

otherwise.

This automaton is non-deterministic: for each state not currently working on a constraint, `generate` defines one outgoing transition for every token in our vocabulary, while `start` defines distinct transitions for some tokens. Thus, the construction we present in Section 2.2 cannot be directly applied to this automaton. This is not a fundamental difficulty, but rather a notational mismatch. In fact, the only change that needs to be made is to modify our definition of the depth-wise hypothesis sets to

$$H^t(d) = \{\boldsymbol{h} \mid \boldsymbol{h} \in H^t, \exists q_i : q_i \in \delta^*(q_0, \boldsymbol{h}) \wedge d(q_i) = d\}$$

in order to account for the automaton's transition function δ, and consequently δ^*, being set-valued instead of state-valued.

Conceptually, this does change the picture, in that hypotheses can now exist in multiple beams, rather than just one. Concretely, when generating the first token of a constraint, this can either be done via the `start` transition (in which case that token is "counted" as the start of a constraint, or via the `generate` transition (in which case it isn't counted), leading to the same token sequence appearing as hypotheses for two distinct depth values. While retaining two copies of the same hypothesis might leave less room in a beam for other hypotheses, we do not expect this to be very common or consequential in practice.