Towards Consistent Language Models Using Controlled Prompting and Decoding

Jasmin Mousavi
Oregon State University
Corvallis, Oregon, USA
mousavij@oregonstate.edu

Arash Termehchy
Oregon State University
Corvallis, Oregon, USA
termehca@oregonstate.edu

ABSTRACT

Large language models (LLMs) have shown unprecedented abilities in generating linguistically coherent and syntactically correct output. However, they often return incorrect and inconsistent answers to input questions. Due to the complexity and uninterpretability of the internally learned representations, it is challenging to modify LLMs such that they provide correct and consistent results. To address this challenge, recent research has focused on controlling the outputs of LLMs through methods like constrained optimization and probabilistic inference. While these approaches mark significant progress, they have limitations in terms of usability, efficiency, and linguistic coherence. Some methods require extensive fine-tuning, making them less practical for general use, while others compromise the linguistic quality of the output. To address these limitations, we explore adding constraints to the prompt. Our experimental findings reveal that this approach reduces the need for fine-tuning and enhances the generation quality, leading to improvements in efficiency and linguistic coherence of the generated output.

ACM Reference Format:
Jasmin Mousavi and Arash Termehchy. 2024. Towards Consistent Language Models Using Controlled Prompting and Decoding. In *Proceedings of Data Management for End-to-End Machine Learning (DEEM '24).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Large language models (LLMs) have shown unprecedented abilities in processing natural languages [9]. They effectively generalize to perform various tasks with few or no training examples. Thus, there is a rapidly growing interest in using them to solve data-driven problems, such as, interactive question answering.

Nonetheless, LLMs often provide incorrect answers to input queries and perform inaccurate inferences [9]. Several studies indicate the recent LLMs provide up to 40% erroneous answers to factual questions [9]. These erroneous results are important obstacle for use of LLMs in real-world applications.

To address the problem of inaccurate answers returned by LLMs, we should recognize that **LLMs are not knowledge bases, but rather probabilistic or approximate models of factual information**. LLMs may over-generalize patterns and relationships observed in the sub-sequences of pretraining data, which might lead to returning spurious relationships and inaccurate results. The uninterpretable mixture of linguistic patterns and factual information has made it challenging to eliminate incorrect information.

One approach is to augment LLMs with *additional and potentially relevant information* from external data sources [6, 8], i.e., retrieval-based LLMs.These methods often add extra information to the context considered during pretraining. This line of research have improved the accuracy of LLMs to a limited degree, as it does not address the core issue of having spurious and incorrect information in LLMs. It is unclear whether adding more relevant information eliminate inaccurate information stored in the model. Moreover, finding sufficiently many relevant data sources, particularly for long-tail entities, may pose challenges.

It is challenging to ensure that an LLM learns accurate generalizations and returns correct answers as it may require perfect knowledge of unobserved data. Even with perfect knowledge of unobserved data, it is challenging to guarantee that LLMs learn accurate generalizations and returns correct answers. Nevertheless, we may be able to restrict its decoding to **adhere to declarative constraints** in the domain to avoid generating incorrect results.

Constraints are essentially rules or guidelines that govern the behavior or output of a system. They can be defined by human experts or learned from data in an unsupervised manner [2, 10]. Compared to retrieval-based augmentation, we argue that **constraints offer a more robust and adaptable framework for reducing inconsistencies in LLMs**. There are two key advantages to using constraints. First, their ability to encapsulate rules governing the underlying domain enables a system to generalize beyond particular instances in a dataset, i.e., out of distribution generalization. Second, that constraints are a form of high-level knowledge, effectively abstracting large quantities of data. Their compressed representation offers a flexible and efficient method of augmenting LLMs by (1) allowing for soft incorporation of constraints (e.g. adhere to a constraint with 80% probability), (2) reducing the size of information used as context to LLMs, and (3) providing a structured way to control the output of LLMs.

There has been recent effort on limiting the output of LLMs so they *follow given constraints*, e.g., contain certain keywords [3, 7]. These methods use constrained optimization or probabilistic inference over the sequences generated by the LLM to reduce the probability of the outputs with invalid patterns. These efforts are steps in the right direction but fall short of ensuring usable, scalable, and linguistically coherent outputs from LLMs. For instance, NeuroLogic [7] requires task-specific fine-tuning, which is impractical as

DEEM '24, June 9, 2024, Santiago, Chile
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

LLMs grow in size. On the other hand, Sequential Monte Caro [3] is compatible with off-the-shelf models, but often fails to maintain linguistic coherence due to its simple masking decoding strategy. Both methods, applied only during decoding, don't address the LLM's potential to learn and represent spurious relationships. This is hard to control due to the difficulty in interpreting LLMs' learned representations. For instance, the learned spurious relationship about one entity might impact how an LLM answers a question about a different but related entity.

To overcome these limitations, we augment the prompt with domain-specific constraints. Prompting with constraints offers three advantages. First, it leverages LLMs' in-context learning capabilities, thereby reducing the need for fine-tuning. Second, it can introduce domain knowledge not present in the training data, e.g., each patient is a human. Hence, the modified prompt might convey more information about the domain than the original one. Third, it expresses the properties of entities that are consistent with constraints in the domain. Moreover, consistent answers depend on the context of the domain constraints, i.e., different domains require different lines of reasoning. By incorporating these constraints into the input context, LLMs can generate higher quality output distributions, enabling the decoder to work more effectively.

In this paper, we explore the use of constraints within prompts to improve the limitations of NeuroLogic [7] and Sequential Monte Carlo [3] decoding strategies. We conduct empirical results for integrating constraints in Llama-2 [12] on the CommonGen benchmark [4], without fine-tuning (Section 5). We identify and discuss the trade-offs between generation quality, constraint satisfaction, and efficiency. We find that optimizing all these aspects is not possible by just adding constraints to the prompt or decoder alone. However, combining constraints in both prompting and decoding, shows improvement. Specifically, compared to using only decoder strategies, adding constraints to prompts leads to improvements in efficiency and generation quality. These results underscore the effectiveness of end-to-end strategies, where prompts and decoders work together in addressing inconsistencies in LLMs.

2 BACKGROUND

Constraints. In our problem, a constraint is defined over the sequence of tokens, i.e., words, generated by an LLM. Given a generated sequence of words S, let us define an indicator function $I(w_j, S)$ that returns true if a word w_j occurs in S.

A set of constraints can be formulated in Conjunctive Normal Form (CNF) as a conjunction ($\wedge$) of clauses $\wedge_i^n C_i$, where each clause C_i is a disjunction ($\vee$) of literals:

$$\underbrace{(I(w_{11}, S) \vee \cdots \vee I(w_{1k_1}, S))}_{C_1} \wedge \cdots \wedge \underbrace{(I(w_{n1}, S) \vee \cdots \vee I(w_{nk_n}, S))}_{C_n}$$

where each constraint $I(w_j, S)$ represents a literal.

For example, suppose we would like to generate a sentence that uses the concepts from the set of keywords x = [dog, run, field]. Therefore, the objective is to generate an output sequence S that contains all keywords in x or its inflections (e.g., dog = [dog, dogs, dogging, dogged]). This expressed in CNF is: $(I(\text{dog}, S) \vee I(\text{dogs}, S) \vee I(\text{dogging}, S) \vee I(\text{dogged}, S)) \wedge (I(\text{run}, S) \vee I(\text{runs}, S) \vee I(\text{running}, S) \vee I(\text{ran}, S)) \wedge (I(\text{field}, S) \vee I(\text{fields}, S))$.

Constraint Satisfaction. Given a set of constraints C expressed in CNF and sequence S, constraint satisfaction refers to the process of checking if sequence S violates constraints in C. This process is gauged using two key metrics: coverage and satisfaction.

Coverage is a number between 0 and 1, calculated as the proportion of clauses in C that evaluate to true: $\frac{1}{|C|}\sum_{i=1}^{|C|} C_i$ where $|C|$ is the number of clauses in C. *Satisfaction* is a Boolean assignment (0 or 1) that assesses whether S adheres to *all* clauses in C: $\wedge_{i=1}^{|C|} C_i$.

3 CURRENT CONSTRAINED DECODING STRATEGIES

We leverage two decoding strategies with varying levels of satisfaction: *soft* constraint decoding with NeuroLogic [7] and *hard* constraint decoding with Sequential Monte Carlo [3].

NeuroLogic (NL) [7] is an inference time decoding algorithm that uses a variant of beam-search. The objective is to optimize the probability of generating sequences while also steering towards constraints using a penalty term. Due to the interest of using off-the-shelf models, we chose *not to fine-tune* an LLM for using the NeuroLogic decoder. It is important to note, however, their experiments were conducted using a fine-tuned model.

Sequential Monte Carlo (SMC) [3] is an inference time masked decoding algorithm. They model sequence generation as a probabilistic inference problem using a variant of Sequential Monte Carlo with particle filtering. In SMC, a user writes a program that specifies the desired constraints in a sequential manner. The user may also specify the number of particles used, where each particle acts as a weighted sample of the posterior distribution. We programmed constraints, i.e. contains certain keywords, as an infilling problem, where keywords are sampled with a masked vocabulary.

4 PROMPTING WITH CONSTRAINTS

Integrating structured data, such as constraints, into prompts poses challenges for LLMs, as they have been trained on unstructured data and impose restrictions on context length, e.g., 4096 tokens for Llama-2 [12]. It is unclear whether the LLM's low-dimensional representation can accurately and reliably reflect these domain-specific constraints. Hence, it is challenging to ensure constraint satisfaction solely through prompt augmentation. Nonetheless, prompting with constraints can still bias the output distribution, allowing the decoder to work more effectively.

For input prompt P = "write a sentence", we construct two prompting techniques for constraints with varying representations.

Conjunctive Normal Form (CNF) prompting style models constraints, i.e., keywords, in conjunctive normal form. For example, the keywords [dog, run, field] in conjunctive normal form is (dog $\vee$ dogs $\vee$ ...) $\wedge$ (run $\vee$ running $\vee$ ...) $\wedge$ (field $\vee$ fields). We can translate this constraint to text by converting $\vee$ to *or* and $\wedge$ to *and*. Hence, our final prompt is "Write a sentence using the words (dog or dogs or ...) and (run or running or ...) and (field or fields)".

Abstract (ABS) prompting style describes an abstract instance of a constraint, e.g., "Given a set of words x, write a sentence using all words in x or inflections of x". Since *ABS* prompts do not include specific instances of keyword inflections, it is more compressed than *CNF* prompts.

4.1 Prompting and Decoding with Constraints

Augmenting either the prompt or the decoder in isolation can help reduce inconsistencies in LLMs, but each approach has its drawbacks. While decoders can offer satisfaction guarantees, they may slow down inference and reduce generation quality due to significant shifts in output distribution. In contrast, prompting maintains generation quality but lacks the ability to ensure constraint satisfaction. To overcome the limitations of each method individually, we propose exploring an end-to-end approach, integrating constraints in both prompting and decoding. Additionally, we aim to investigate how various prompting techniques impact the effectiveness of decoder strategies.

5 EMPIRICAL RESULTS

In this section we present our empirical results for integrating constraints with LLMs using the CommonGen benchmark [4]. We identify the risks and trade-offs of augmenting LLMs with constraints for the *prompt only* (Section 4) and *decoder only* (Section 3) strategies, in terms of generation quality, constraint satisfaction, and time. We also explore whether injecting constraints into both *prompt + decoder* (Section 4.1), will help or hurt any risks and trade-offs that exist in the prompt or decoder alone.

5.1 Experimental Setup

LLM. We use Llama-2 [12] as our pretrained language model across all experiments.

Dataset. The CommonGen dataset [4] is a benchmark designed for controlling language model generation with constraints, i.e., contain certain keywords. Given a set of keywords, the goal is to generate a sentence using all the keywords or the infections of the keywords. Each set contains 3-5 keywords. The dataset is split into train, validation, and test sets. Typically, those using the CommonGen dataset would first fine-tune their model using the training set. However, we focus on using inference-based algorithms that can be used with off-the-shelf models *without fine-tuning*. Our results are conducted over the test set.

In-Context Examples. We supply the prompt with additional in-context examples, i.e., *n-shot*, extracted from the training set.

Constraints. In CommonGen, constraints can be defined for a set of keywords $[w1, w2, w3]$ as follows. If S is a sentence, then S must contain $w1$ or one of its inflections, $w2$ or one of its inflections, and $w3$ or one of its inflections. The objective is to generate sentences that adhere to this constraint. A key characteristic of this constraint is its allowance for multiple valid outputs, stemming from the underspecificity of the input. This leads to a wide array of possible sentences that represent instances of the constraint.

Metrics. Generation Quality is measured using automatic metrics, such as ROUGE [5], BLEU [11], CIDEr [13], and SPICE [1]. These metrics generate a quality score for the generated sentence based on human generated reference sentences, where a perfect score is 100. Constraint Satisfaction measures the method's ability to fully satisfy the constraint (used all keywords or their inflections), i.e., *satisfied*. We also calculate *coverage*, which is an average over the percentage of keywords (or their inflections) used in the generated sequence. Time is computed as the time taken (in seconds) for generating a sequence, i.e., inference time.

5.2 Results & Analysis

Results over all experiments can be found in Table 1.

Prompt Only. We aim to understand how varying constraint representation, i.e., *ABS* vs. *CNF*, and in-context examples, i.e., *n-shot*, impact generation quality, constraint satisfaction, and time.

Across most experiments *ABS* prompting achieves higher satisfaction than *CNF* prompting. This suggests that LLMs can understand abstract, high-level descriptions of constraints. Given the fact that *CNF* prompts include all the inflections, one would expect higher constraint satisfaction across all experiments, however, this is not the case. With the exception of *CNF 1-shot*, *ABS* style prompting obtains higher satisfaction than *CNF*. *ABS* prompting outperforms *CNF* prompting in terms of generation quality across all experiments. *CNF* style prompts are inherently more structured and further from 'natural language' compared to *ABS* style prompts. This suggests structured prompts are less beneficial and may require a fine-tuning strategy.

Increasing input length does not have significant impacts on inference time. Despite *ABS* prompts having a smaller constraint representation size than *CNF* prompts, there is little change in inference time across all *n-shot* experiments. In-context examples boosts quality in both prompting strategies, but hurts satisfaction in *CNF 2-shot*. Including more than one in-context example worsens constraint satisfaction for *CNF* style prompts. This suggests that extending the input context with inflections for every in-context example may lead to noisy, sub-optimal distributions for generation.

Decoder Only. In this section we discover the impacts of the output layer, i.e., decoder, on generation quality, constraint satisfaction, and time. We compare two decoding strategies: soft constraint decoding, i.e., beam-based *NL* and hard constraint decoding, i.e., masked-based *SMC*.

In the absence of fine-tuning, beam-based/soft constraint decoding, i.e., NL, encounters challenges in both generation quality and constraint satisfaction. Compared with SMC, NL is more dependent on a high quality output distributions. This suggest that soft constraint decoding requires higher quality output distributions.

Increasing the number of particles for SMC decoding does not yield quality or satisfaction improvements while increasing inference time. This observation indicates that the underlying distribution may be of low quality, as increasing the number of particles does not enhance performance. Moreover, in cases of uncertainty, the decoder will not see benefits by increasing computational resources. Although the SMC decoder achieves 100% constraint satisfaction, this achievement comes at the cost of significantly increased inference time. For example, the longest inference time recorded among the tested prompting strategies was only 1.85 seconds, in contrast, SMC with 8 particles required a considerably longer duration of 22.92 seconds for generation. This indicates a substantial increase in computational time required to achieve complete constraint satisfaction with masked decoding strategies, such as SMC.

Despite the improvements in constraint satisfaction, decoder only strategies degrade generation quality and increase time.

Prompt & Decoder. Although prompt only strategies have higher performance on generation quality and time, they cannot provide any guarantees on constraint satisfaction. Conversely, decoding strategies optimize over constraint satisfaction, but at the cost of

Method	*Generation Quality*				*Constraint Satisfaction*		*Time*
	ROUGE-L	BLEU-4	CIDEr	SPICE	Coverage	Satisfied	Seconds
Prompt Only							
ABS 0-shot	25.37	06.29	04.34	13.81	51.61	18.84	01.56
ABS 1-shot	29.46	08.41	06.22	18.49	74.49	35.34	01.85
ABS 2-shot	**31.34**	**10.60**	**07.33**	**20.06**	76.74	38.74	01.83
CNF 0-shot	22.82	03.74	02.37	12.13	42.17	11.09	01.84
CNF 1-shot	29.60	08.07	05.88	18.77	**77.47**	**38.88**	01.51
CNF 2-shot	30.93	09.92	06.81	19.22	74.48	34.34	01.46
Decoder Only							
NL [7], beam=8	10.05	00.00	00.08	02.67	02.41	00.00	03.46
NL [7], beam=32	10.36	00.00	00.06	02.31	01.12	00.00	12.47
NL [7], beam=64	9.74	00.23	00.04	02.59	00.96	00.00	24.01
SMC [3], particle=8	23.10	02.60	**01.71**	15.37	**100.0**	**100.0**	22.92
SMC [3], particle=16	22.86	02.52	01.62	**15.55**	**100.0**	**100.0**	22.96
SMC [3], particle=32	**22.92**	**02.64**	01.69	15.26	**100.0**	**100.0**	23.17
Prompt & Decoder							
ABS 0-shot + NL [7], beam=8	36.54	14.82	10.72	20.65	95.93	83.43	05.30
ABS 1-shot + NL [7], beam=8	39.07	19.25	12.13	23.25	94.13	76.55	04.61
ABS 2-shot + NL [7], beam=8	39.39	19.76	12.26	23.65	93.81	75.48	05.11
CNF 0-shot + NL [7], beam=8	15.41	03.98	01.56	06.42	09.38	01.00	07.47
CNF 1-shot + NL [7], beam=8	39.66	**25.73**	**13.30**	**24.18**	79.91	39.08	09.37
CNF 2-shot + NL [7], beam=8	**39.81**	25.35	12.84	23.57	75.49	27.99	11.30
ABS 0-shot + SMC [3], particle=8	25.86	04.00	02.79	18.80	**100.0**	**100.0**	25.33
ABS 1-shot + SMC [3], particle=8	27.86	05.66	04.46	20.27	**100.0**	**100.0**	25.14
ABS 2-shot + SMC [3], particle=8	28.62	06.17	04.90	20.55	**100.0**	**100.0**	29.96
CNF 0-shot + SMC [3], particle=8	26.27	04.07	03.14	19.85	**100.0**	**100.0**	27.51
CNF 1-shot + SMC [3], particle=8	27.40	04.65	03.84	20.29	**100.0**	**100.0**	34.10
CNF 2-shot + SMC [3], particle=8	28.44	05.93	04.54	20.70	**100.0**	**100.0**	48.76

Table 1: Performance results on generation quality, constraint satisfaction, and time over the CommonGen test set for different generation methods: *decoder only*, *prompt only*, and *prompt + decoder*. With the exception of time, a perfect score is 100.

generation quality and time. In this section we aim answer whether the prompt and decoder can work together to improve the disadvantages of using the prompt or decoder alone, i.e., an end-to-end system. More specifically, we would like to understand how different strategies work together and whether they induce any trade-offs between our metrics.

Augmenting the prompt with constraints enhances generation quality and constraint satisfaction, indicating that prompting results in a higher-quality output distribution for the decoder to operates on. Notably, the *NL* decoder, although underperforming as a standalone decoder, shows remarkable improvement in quality when combined with prompts. This demonstrates that soft constraint decoder performance depends on the quality of the output distribution. Although prompting improves quality in *SMC*, it has significant impacts on time. Checking for hard constraints within the *SMC* decoding strategy is less scalable when compared to the implementation of soft constraints in *NL*. In contrast, the *NL* decoder benefits in both quality and time. Due to the higher quality output distributions produced with prompting, the *NL* decoder spent less time searching, leading to reductions in inference time.

Across most experiments, the *NL* decoder achieves higher generation quality than *SMC*. The use of soft constraints in *NL* results in less drastic distribution changes compared to *SMC*, allowing for higher quality generation, albeit with a trade-off in constraint satisfaction. Despite the *NL* decoder leveraging CNF formula, it exhibits higher satisfaction levels with *ABS* style prompts. This indicates that structured prompts could potentially limit the model's performance by producing sub-optimal output distributions for the decoder. It suggests that high-level concepts and relationships might be more effective inputs to the model when optimizing the output distribution for decoding.

REFERENCES

[1] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*. Springer, 382–398.

[2] Sridevi Baskaran, Alexander Keller, Fei Chiang, Lukasz Golab, and Jaroslaw Szlichta. 2017. Efficient discovery of ontology functional dependencies. In *Proceedings of the ACM on Conference on Information and Knowledge Management*.

[3] Alexander K. Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K. Mansinghka. 2023. Sequential Monte Carlo Steering of Large Language Models using Probabilistic Programs. arXiv:2306.03081 [cs.AI]

[4] Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2019. CommonGen: A constrained text generation challenge for generative commonsense reasoning. *arXiv:1911.03705* (2019).

[5] Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 human language technology conference of the North American chapter of the association for computational linguistics*. 150–157.

[6] Qi Liu, Dani Yogatama, and Phil Blunsom. 2022. Relational memory-augmented language models. *Transactions of the Association for Computational Linguistics* 10 (2022), 555–572.

[7] Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Neurologic decoding:(un) supervised neural text generation with predicate logic constraints. *arXiv preprint arXiv:2010.12884* (2020).

[8] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented Language Models: a Survey. arXiv:2302.07842 [cs.CL]

[9] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[10] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. 2015. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment* 8, 10 (2015), 1082–1093.

[11] 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*. 311–318.

[12] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[13] R Vedantam, C Lawrence Zitnick, and D Parikh. [n. d.]. Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4566–4575.