Tractable Control for Autoregressive Language Generation

Honghua Zhang* Meihua Dang* Nanyun Violet Peng Guy Van den Broeck
University of California, Los Angeles

Oral Presentation

Prompting is Not All You Need

Generate a sentence using "frisbee", "caught" and "dog", following the given order.

After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.

Here's the correct sentence: The dog caught the frisbee in mid-air, showing off its amazing catching skills.

ChatGPT fails to follow simple logical constraints!

Constrained Generation is Challenging

Logical Constraint α: e.g., text contains keyword "*winter*"

Constrained Autoregressive Generation. Our goal is to generate from

$$p(x_{1:n} \mid \alpha) = \prod_t p(x_{t+1} \mid x_t, \alpha)$$

at each step, e.g., suppose we have generated the first t tokens $x_{1:t}$ as "*the weather is*"; then we generate the next token x_{t+1} from

$$p(x_{t+1} \mid x_{1:t}, \alpha) \propto \boxed{p(x_{t+1} \mid x_{1:t})} \cdot \boxed{p(\alpha \mid x_{t+1}, x_{1:t})}$$

x_{t+1}	$p_{lm}(x_{t+1} \mid x_{1:t})$
cold	0.05
warm	0.10

Off-the-shelf LM distribution.

x_{t+1}	$p_{lm}(\alpha \mid x_{t+1}, x_{1:t})$
cold	?
warm	?

Steers LM to satisfy α. Intractable for LLMs.

Requires **marginalization** over all suffixes $x_{t+1:n}$ containing "winter".

GeLaTo

We propose **GeLaTo** (**Ge**nerating **La**nguage with **T**ractable C**o**nstraints) to guide autoregressive generation from LLMs.

Tractable Probabilistic Models (TPMs) are generative models $p_{tpm}(x_{1:n})$ that allow efficient conditioning. We use **hidden Markov models** (HMMs) as an example.

Step 1. Distilling an HMM from LM

- Train p_{hmm} on $D \sim p_{lm}$ to minimize their KL-divergence.

Step 2. Probabilistic Reasoning with Constraints

- Compute $p_{hmm}(\alpha \mid x_{1:t}, x_{t+1})$ to approximate $p_{lm}(\alpha \mid x_{1:t}, x_{t+1})$; then generate from:

$$p_{gelato}(x_{t+1} \mid x_{1:t}, \alpha) \propto p_{lm}(x_{t+1} \mid x_{1:t}) \cdot p_{hmm}(\alpha \mid x_{t+1}, x_{1:t})$$

*GeLaTo can also help prompting!

Given some prompt π that represents α, e.g., "keywords = XXX", we can combine p_{hmm} and p_{lm} by taking their weighted geometric mean:

$$p_{gelato}(x_{t+1} \mid x_{1:t}, \pi, \alpha) \propto p_{lm}(x_{t+1} \mid x_{1:t}, \pi)^{1-w} \cdot p_{hmm}(x_{t+1} \mid x_{1:t}, \alpha)^{w}$$

*GeLaTo can enforce various logical constraints

1. Keywords appear (in any order/form of inflections)
2. (Some) keywords are generated following a specific order.
3. (Some) keywords must appear at specified positions.
4. (Some) keywords must not appear in the generated text.

Advantages of GeLaTo

1. Logical constraint α is **guaranteed** to be satisfied.
   - When generating next token x_{t+1}, if $x_{t+1} = w$ would make α unsatisfiable, then $p_{gelato}(x_{t+1} \mid \alpha, x_{1:n}) = 0$; hence w would not be generated.
2. The training of p_{hmm} does not depend on α, which is only imposed during generation. Once p_{hmm} is trained, GeLaTo **generalizes to any tractable constraints**.

Probabilistic Reasoning with Constraints

Consider a logical constraint α encoded as:

$$(w_{1,1} \vee \ldots \vee w_{1,d_1}) \wedge \ldots \wedge (w_{m,1} \vee \ldots \vee w_{m,d_m})$$

each w_{ij} is a a string of tokens ("keywords") that must appear

$$\alpha = (\text{"swims"} \vee \text{"like swimming"}) \wedge (\text{"lake"} \vee \text{"pool"})$$

Need to compute $p_{hmm}(\alpha \mid x_{1:t}, x_{t+1})$ *to enforce* α.

Efficient Probabilistic Reasoning for HMMs

HMMs define distributions over $x_{1:n}$ and latent variables $z_{1:n}$:

$$p(x_{1:n}, z_{1:n}) = \prod_t p(z_{t+1} \mid z_t) p(x_t \mid z_t)$$

Assume α only contains single-token keywords, then we can compute $p(\alpha_{t:n})$, as well as $p(\alpha_{1:n}, x_{1:t})$, by

$$p(\alpha_{t:n} \mid z_t) = \sum_{w \in vocab} \sum_{z_{t+1}} p((\alpha \backslash w)_{t+1:n} \mid z_{t+1}) p(z_{t+1} \mid z_t) p(x_t = w \mid z_t)$$

The time complexity for sampling from $p_{gelato}(x_{1:n} \mid \alpha)$ *is* $O(2^m n)$

Experiments

Commonsense Generation (CommonGen)

Input Concepts: snow, car, drive
Output 1: The car drives down a snow-covered road.
Output 2: Driving through the snow, the car crashed.

Method	*Generation Quality* ROUGE-L		BLEU-4		CIDEr		*Constraint* Success Rate	
Unsupervised	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*
InsNet	-	-	18.7	-	-	-	**100.0**	-
NeuroLogic	-	41.9	-	24.7	-	14.4	-	-
A*esque	-	**44.3**	-	28.6	-	**15.6**	-	-
NADO	-	-	26.2	-	-	-	-	-
GeLaTo	**44.3**	43.8	**30.3**	**29.0**	**15.6**	15.5	**100.0**	**100.0**
Supervised	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*
NeuroLogic	-	42.8	-	26.7	-	14.7	-	93.9†
A*esque	-	43.6	-	28.2	-	15.2	-	97.9†
NADO	44.4†	-	30.8	-	16.1†	-	88.8†	-
GeLaTo	**46.2**	**45.9**	**34.0**	**34.1**	**17.2**	**17.5**	**100.0**	**100.0**

Table 1. Automatic evaluation results on CommonGen.

Method	Concepts	Plausibility	Quality	Overall
GPT2	2.47	**2.52**	2.65	2.28
NADO	**2.71**	**2.54**	**2.73**	2.54
GeLaTo	**2.73**	**2.52**	**2.70**	**2.60**

Table 2. Human evaluation results on CommonGen (supervised setting).

Run-time Comparison

# of concepts	3	4	5
Unsupervised			
A*esque	472.9	542.5	613.9
GeLaTo (16)	13.5 ± 4.4	21.9 ± 5.37	39.3 ± 6.3
GeLaTo (128)	69.8 ± 32.3	97.9 ± 39.5	143.0 ± 44.4
Supervised			
A*esque	8.5	9.6	11.4
GPT2 (16)	5.8 ± 1.1	13.0 ± 1.6	29.3 ± 3.2
GPT2 (128)	9.4 ± 1.8	21.1 ± 11.9	33.7 ± 3.5
GeLaTo (16)	11.1 ± 2.8	22.0 ± 5.0	41.6 ± 5.6
GeLaTo (128)	49.8 ± 20.8	88.7 ± 30.5	127.6 ± 30.4

Table 3. Time (seconds) for generating one sentence on CommonGen.

Yelp!Review and *News*: fixing order of keywords

The *Yelp!Review* and *News* datasets are similar to CommonGen except for that they require keywords to appear in the given order.

Method \ Dataset	Yelp!Review	News
InsNet	5.8	5.0
NADO	6.0	4.5
GeLaTo	**6.6**	**5.4**

Table 4. BLEU-4 scores for Yelp!Review and News datasets

Paper → Code →

ICML 2023

References: [1] Lu, S., Meng, T., and Peng, N. Insnet: An efficient, flexible, and performant insertion-based text generation model. In NeurIPS, 2022.
[2] Lu, X., West, P., Zellers, R., Le Bras, R., Bhagavatula, C., and Choi, Y. Neurologic decoding:(un) supervised neural text generation with predicate logic constraints. In Proceedings of NAACL, 2021.
[3] Lu, X., Welleck, S., West, P., Jiang, L., Kasai, J., Khashabi, D., Le Bras, R., Qin, L., Yu, Y., Zellers, R., Smith, N. A., and Choi, Y. NeuroLogic a*esque decoding: Constrained text generation with lookahead heuristics. In Proceedings of NAACL, 2022.
[4] Meng, T., Lu, S., Peng, N., and Chang, K.-W. Controllable text generation with neurally-decomposed oracle. In NeurIPS, 2022.

Acknowledgements: This work was funded in part by the DARPA Perceptually-enabled Task Guidance (PTG) Program under contract number HR00112220005, NSF grants #IIS-1943641, #IIS-1956441, #CCF-1837129, an SRA from Meta, a research gift from Amazon Alexa AI, and a gift from RelationalAI. GVdB discloses a financial interest in RelationalAI.