UCLA Computer Science

STAR AI RESEARCH LAB UCLA

Symbolic Reasoning for Large Language Models

Guy Van den Broeck

Workshop on the Integration of Large Language Models and Reasoning - Apr 19 2024

Outline

1. The paradox of learning to reason from data

   ~~*end-to-end learning*~~

2. Symbolic reasoning at generation time

3. Symbolic reasoning at training time

   logical + probabilistic reasoning + deep learning

Outline

1. **The paradox of learning to reason from data**

   ~~end-to-end learning~~

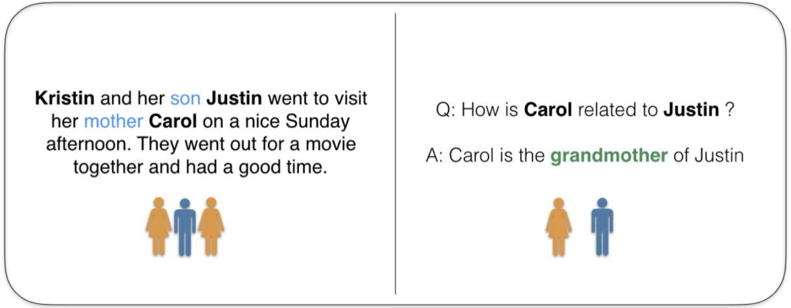
2. Symbolic reasoning at generation time

3. Symbolic reasoning at training time

   logical + *probabilistic reasoning* + *deep learning*

Can Language Models Perform Logical Reasoning?

Language Models achieve high performance on “reasoning” benchmarks.



Reasoning Example from the CLUTRR dataset

Unclear whether they follow the rules of logical deduction.

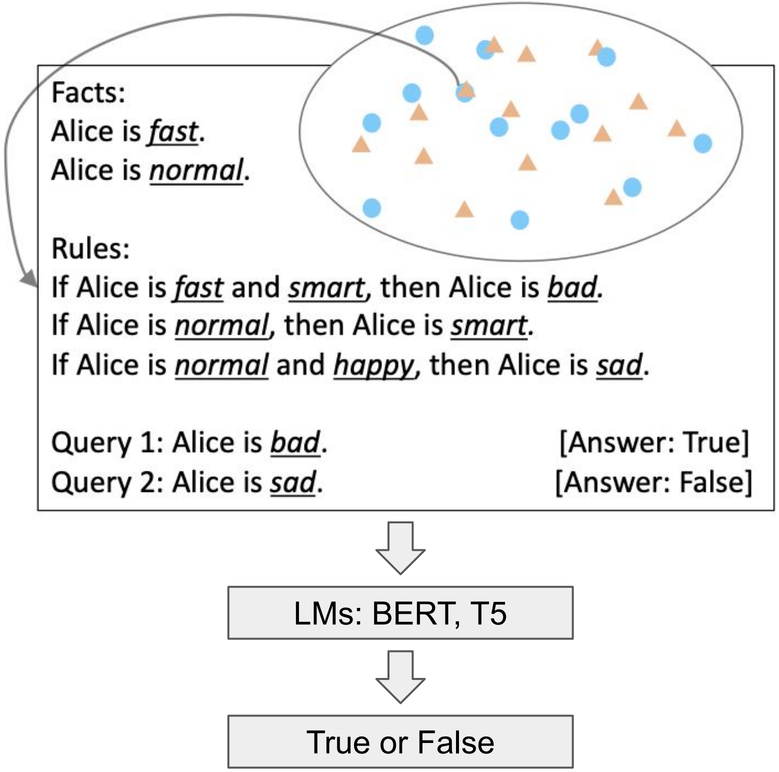
Language Models:
input → ? → Carol is the grandmother of Justin.

Logical Reasoning:
input → Justin in Kristin’s son; Carol is Kristin’s mother; → Carol is Justin’s mother’s mother; if X is Y’s mother’s mother then X is Y’s grandmother → Carol is the grandmother of Justin.

Problem Setting: SimpleLogic

Easiest of reasoning problems:

1. **Propositional logic** fragment
   Bounded vocabulary & number of rules
   & reasoning depth – finite space (≈ 10^360)
2. **No language variance**: templated language
3. **Self-contained**
   No prior knowledge
4. **Purely symbolic** predicates
   No shortcuts from word meaning
5. **Tractable** logic (definite clauses)
   Can always be solved efficiently



Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

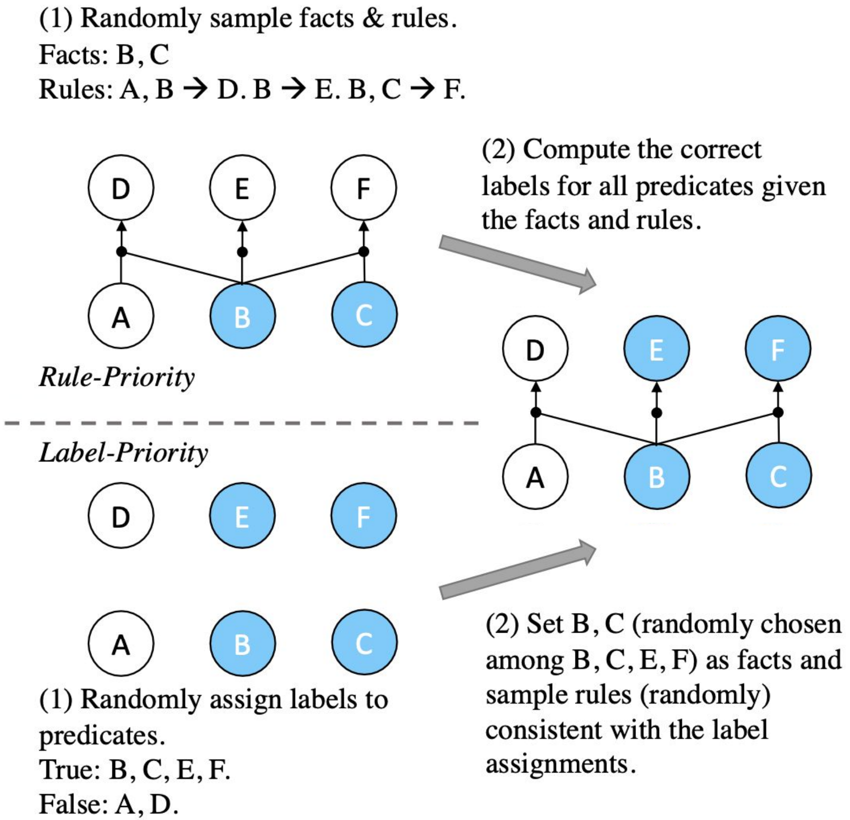
SimpleLogic

Generate textual train and test examples of the form:

Rules: If witty, then diplomatic. If careless and condemned and attractive, then blushing. If dishonest and inquisitive and average, then shy. If average, then stormy. If popular, then blushing. If talented, then hurt. If popular and attractive, then thoughtless. If blushing and shy and stormy, then inquisitive. If adorable, then popular. If cooperative and wrong and stormy, then thoughtless. If popular, then sensible. If cooperative, then wrong. If shy and cooperative, then witty. If polite and shy and thoughtless, then talented. If polite, then condemned. If polite and wrong, then inquisitive. If dishonest and inquisitive, then talented. If blushing and dishonest, then careless. If inquisitive and dishonest, then troubled. If blushing and stormy, then shy. If diplomatic and talented, then careless. If wrong and beautiful, then popular. If ugly and shy and beautiful, then stormy. If shy and inquisitive and attractive, then diplomatic. If witty and beautiful and frightened, then adorable. If diplomatic and cooperative, then sensible. If thoughtless and inquisitive, then diplomatic. If careless and dishonest and troubled, then cooperative. If hurt and witty and troubled, then dishonest. If scared and diplomatic and troubled, then average. If ugly and wrong and careless, then average. If dishonest and scared, then polite. If talented, then dishonest. If condemned, then wrong. If wrong and troubled and blushing, then scared. If attractive and condemned, then frightened. If hurt and condemned and shy, then witty. If cooperative, then attractive. If careless, then polite. If adorable and wrong and careless, then diplomatic. Facts: Alice sensible Alice condemned Alice thoughtless Alice polite Alice scared Alice average
Query: Alice is shy ?

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

Training a transformer on SimpleLogic



Test accuracy for different reasoning depths

Test	0	1	2	3	4	5	6
RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5

Test	0	1	2	3	4	5	6
LP	100.0	100.0	99.9	99.9	99.7	99.7	99.0

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

Has the transformer learned to reason from data?

1. Easiest of reasoning problems (no variance, self-contained, purely symbolic, tractable)
2. RP/LP data covers the whole problem space
3. The learned model has almost 100% test accuracy
4. There exist transformer parameters that compute the ground-truth reasoning function:

   Theorem 1: *For a BERT model with* n *layers and 12 attention heads, by construction, there exists a set of parameters such that the model can correctly solve any reasoning problem in SimpleLogic that requires at most* $n - 2$ *steps of reasoning.*

Surely, under these conditions, the transformer has learned the ground-truth reasoning function!

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

The Paradox of Learning to Reason from Data

Train	Test	0	1	2	3	4	5	6
RP	RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5
	LP	99.8	99.8	99.3	96.0	90.4	75.0	57.3
LP	RP	97.3	66.9	53.0	54.2	59.5	65.6	69.2
	LP	100.0	100.0	99.9	99.9	99.7	99.7	99.0

The BERT model trained on one distribution fails to generalize to the other distribution within the same problem space.

1. If the transformer **has learned** to reason, it should not exhibit such generalization failure.
2. If the transformer **has not learned** to reason, it is baffling how it achieves near-perfect in-distribution test accuracy.

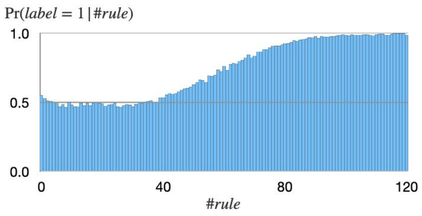
Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

Why? Statistical Features

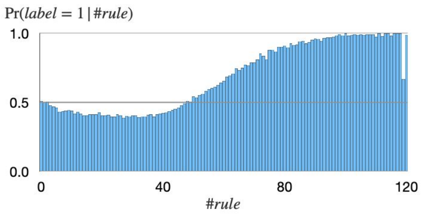
Monotonicity of entailment:
Any rules can be freely added to the axioms of any proven fact.

The more rules given, the more likely a predicate will be proven.

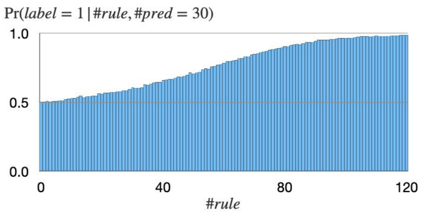
Pr(label = True | Rule # = x) should increase (roughly) monotonically with x



(a) Statistics for examples generated by Rule-Priority (RP).



(b) Statistics for examples generated by Label-Priority (LP).



(c) Statistics for examples generated by uniform sampling;

Model leverages statistical features to make predictions

RP_b downsamples from RP such that Pr(label = True | rule# = x) = 0.5 for all x

Train	Test	0	1	2	3	4	5	6
	RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5
RP	RP_b	99.0	99.3	98.5	97.5	96.7	93.5	88.3

1. Accuracy drop from RP to RP_b indicates that
   the model is using rule# as a statistical feature to make predictions.

2. Potentially countless statistical features

3. Such features are **inherent to the reasoning problem**, cannot make data "clean"

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

First Conclusion

Experiments unveil the fundamental difference between

1. learning to reason, and
2. learning to achieve high performance on benchmarks using statistical features.

Be careful deploying AI in applications where this difference matters.

FAQ: Do bigger transformers solve this problem? No, already 99% accurate…

FAQ: Will reasoning emerge? Perhaps on 99% of human behavior…

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

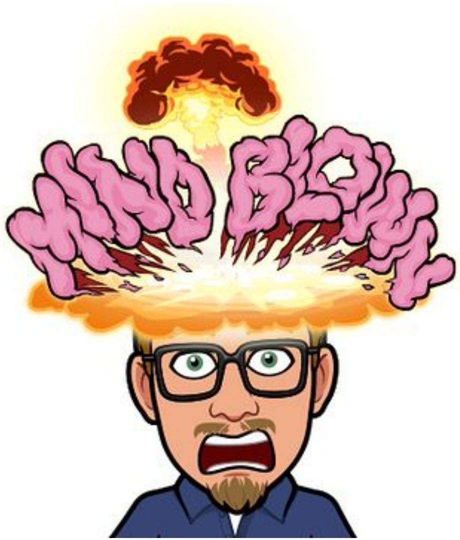
Preview

4. There exist transformer parameters that compute the ground-truth reasoning function:

   Theorem 1: *For a BERT model with n layers and 12 attention heads, by construction, there exists a set of parameters such that the model can correctly solve any reasoning problem in SimpleLogic that requires at most n − 2 steps of reasoning.*

Initialize the model with the perfect parameters that simulate the ground-truth reasoning algorithm.

Then SGD will **un-learn the algorithm** that generalizes OOD and again learn statistical shortcuts.



Outline

1. The paradox of learning to reason from data

   ~~*end-to-end learning*~~

2. **Symbolic reasoning at generation time**

3. Symbolic reasoning at training time

   logical + *probabilistic reasoning* + *deep learning*

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

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.

ChatGPT

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.

ChatGPT

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

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.

ChatGPT

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

A frisbee is caught by a dog.

A pair of frisbee players are caught in a dog fight.

GeLaTo

What do we have?

Prefix: “The weather is”

Constraint α: text contains “winter”

Model only does $p(\text{next-token}|\text{prefix}) =$

cold	0.05
warm	0.10

Train some $q(.\,|\alpha)$ for a specific task distribution $\alpha \sim p_{\text{task}}$

(amortized inference, encoder, masked model, seq2seq, prompt tuning,...)

Train $q(\text{next-token}|\text{prefix}, \alpha)$

What do we need?

Prefix: “The weather is”

Constraint α: text contains “winter”

Generate from $p(\text{next-token}|\text{prefix}, \alpha) =$

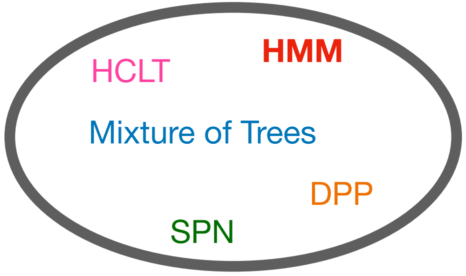
cold	0.50
warm	0.01

$$\propto \sum_{\text{text}} p(\text{next-token}, \text{text}, \text{prefix}, \alpha)$$

Marginalization!

Tractable Probabilistic Models

Probabilistic Circuits



Tractable Probabilistic Models (TPMs) model joint probability distributions and allow efficient probabilistic inference.

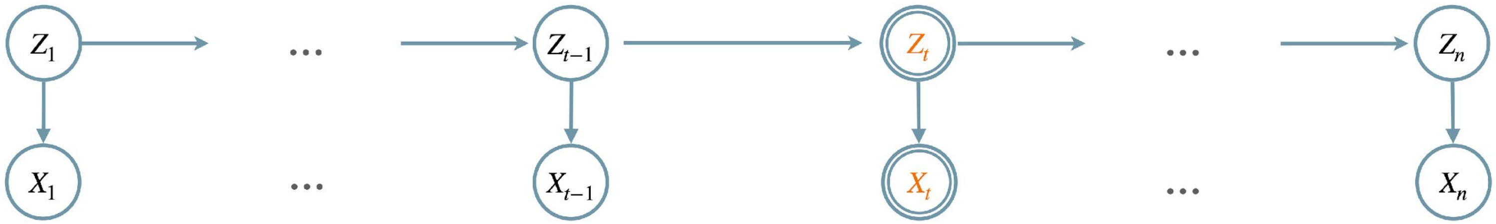
e.g., efficient marginalization:

$$p_{TPM}(\text{3rd token = frisbee, 5th token = dog})$$

For now… keep it simple… just a Hidden Markov Model (HMM)

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Step 1: Distill an HMM p_{hmm} that approximates p_{gpt}



1. HMM with 4096 hidden states and 50k emission tokens
2. Data sampled from GPT2-large (domain-adapted), minimizing KL(p_{gpt} // p_{HMM})
3. Leverages latent variable distillation for training at scale [ICLR 23]. (Cluster embeddings of examples to estimate latent Z_i)

Anji Liu, Honghua Zhang and Guy Van den Broeck. Scaling Up Probabilistic Circuits by Latent Variable Distillation, 2023.

CommonGen: a Challenging Benchmark

Given 3-5 keywords, generate a sentence using all keywords, in any order and any form of inflections. e.g.,

Input: snow drive car

Reference 1: A car drives down a snow covered road.

Reference 2: Two cars drove through the snow.

Constraint α in CNF: $(w_{1,1} \vee \ldots \vee w_{1,d1}) \wedge \ldots \wedge (w_{m,1} \vee \ldots \vee w_{m,dm})$

Each clause represents the inflections for one keyword.

Computing $p(\alpha \mid x_{1:t+1})$

For constraint α in CNF:

$$(w_{1,1} \lor \ldots \lor w_{1,d1}) \land \ldots \land (w_{m,1} \lor \ldots \lor w_{m,dm})$$

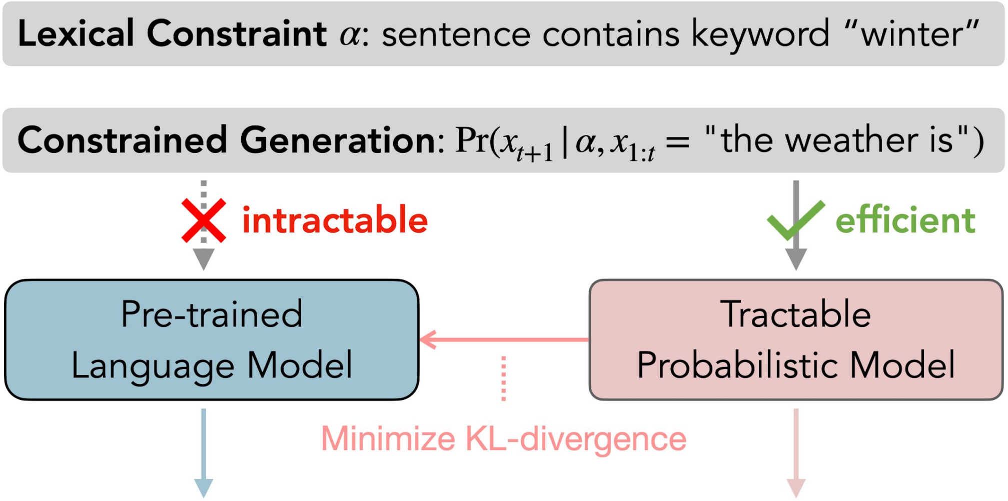
e.g., α = ("swims" ∨ "like swimming") ∧ ("lake" ∨ "pool")

Efficient algorithm:
For m clauses and sequence length n, time-complexity for HMM generation is $O(2^{|m|}n)$

Trick: dynamic programming with clever preprocessing and local belief updates

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

GeLaTo Overview



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

x_{t+1}	$\Pr_{TPM}(\alpha \mid x_{t+1}, x_{1:t})$
cold	0.50
warm	0.01

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

GeLaTo Overview

Lexical Constraint α: sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} \mid \alpha, x_{1:t} = \text{"the weather is"})$

❌ intractable → Pre-trained Language Model

✅ efficient → Tractable Probabilistic Model

Tractable Probabilistic Model → Pre-trained Language Model: Minimize KL-divergence

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

x_{t+1}	$\Pr_{TPM}(\alpha \mid x_{t+1}, x_{1:t})$
cold	0.50
warm	0.01

x_{t+1}	$p(x_{t+1} \mid \alpha, x_{1:t})$
cold	0.025
warm	0.001

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Step 2: Control p_{gpt} via p_{hmm}

Unsupervised

Language model is not
fine-tuned/prompted to satisfy constraints

By Bayes rule:

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

Assume $p_{hmm}(\alpha \mid x_{1:t+1}) \approx p_{gpt}(\alpha \mid x_{1:t+1})$, we generate from:

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

Method	*Generation Quality*								*Constraint Satisfaction*			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
Unsupervised	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*
InsNet (Lu et al., 2022a)	-	-	18.7	-	-	-	-	-	**100.0**	-	**100.0**	-
NeuroLogic (Lu et al., 2021)	-	41.9	-	24.7	-	14.4	-	27.5	-	96.7	-	-
A*esque (Lu et al., 2022b)	-	**44.3**	-	28.6	-	15.6	-	29.6	-	97.1	-	-
NADO (Meng et al., 2022)	-	-	26.2	-	-	-	-	-	96.1	-	-	-
GeLaTo	**44.6**	44.1	**29.9**	**29.4**	**16.0**	**15.8**	**31.3**	**31.0**	**100.0**	**100.0**	**100.0**	**100.0**

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Step 2: Control p_{gpt} via p_{hmm}

Supervised

Language model is fine-tuned to perform constrained generation (e.g. seq2seq)

Empirically $p_{HMM}(\alpha \mid x_{1:t+1}) \approx p_{gpt}(\alpha \mid x_{1:t+1})$ does not hold well enough;

we view $p_{HMM}(x_{t+1} \mid x_{1:t}, \alpha)$ and $p_{gpt}(x_{t+1} \mid x_{1:t})$ as classifiers trained for the same task with different biases; thus we generate from their *weighted geometric mean*:

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

Method	*Generation Quality*								*Constraint Satisfaction*			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
Supervised	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*	*dev*	*test*
NeuroLogic (Lu et al., 2021)	-	42.8	-	26.7	-	14.7	-	30.5	-	97.7	-	93.9†
A*esque (Lu et al., 2022b)	-	43.6	-	28.2	-	15.2	-	30.8	-	97.8	-	97.9†
NADO (Meng et al., 2022)	44.4†	-	30.8	-	16.1†	-	**32.0**†	-	97.1	-	88.8†	-
GeLaTo	**46.0**	**45.6**	**34.1**	**32.9**	**16.7**	**16.8**	31.3	**31.9**	**100.0**	**100.0**	**100.0**	**100.0**

Honghua Zhang, Meihua Dang, Nanyun Peng and Guy Van den Broeck. Tractable Control for Autoregressive Language Generation, 2023.

Advantages of GeLaTo:

1. Constraint α is <u>guaranteed to be satisfied</u>:
   for any next-token x_{t+1} that would make α unsatisfiable, $p(x_{t+1} \mid x_{1:t}, \alpha) = 0$.

2. Training p_{hmm} <u>does not depend on α</u>,
   which is only imposed at inference (generation) time.

3. Can impose <u>additional tractable constraints</u>:
   - keywords follow a particular order
   - keywords appear at a particular position
   - keywords must not appear

Conclusion: you can control an intractable generative model using a tractable probabilistic circuit.

Outline

1. The paradox of learning to reason from data

   ~~*end-to-end learning*~~

2. Symbolic reasoning at generation time

3. **Symbolic reasoning at training time**

   logical + *probabilistic reasoning* + *deep learning*

Neurosymbolic learning of transformers

Given:

1. constraint α (a list of 403 toxic words not to say)
2. training data D

Learn: a transformer Pr(.) that

1. satisfies the constraint α: Pr(α)↑
2. maximizes the likelihood: Pr(D)↑

Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

Neurosymbolic learning of transformer

Given:

1. constraint α (a list of 403 toxic words not to say)
2. training data D

Learn: a transformer Pr(.) that

1. satisfies the constraint α: Pr(α)↑
2. maximizes the likelihood: Pr(D)↑

Pr(α) is computationally hard, even when α is trivial:

What is prob. that LLM ends the sentence with "UCLA"?

Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

Autoregressive distributions are hard…

Pr(α) is computationally hard, even when α is trivial:
What is prob. that LLM ends the sentence with "AAAI"?

Why did it work before?

We were using a separate **tractable proxy** model…

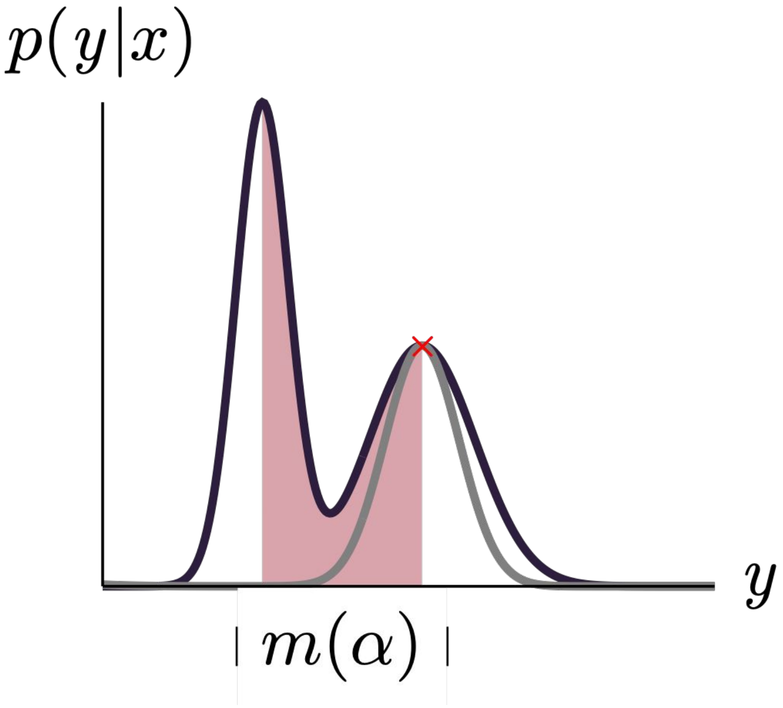
Now we need to train the actual intractable transformer…

Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

Basic Idea:

Use how likely a constraint is to be satisfied around a model sample (x) as a proxy for how likely it is to be satisfied under the entire distribution.

Average over many such samples.



Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

Formally, minimize the *pseudo-semantic loss*

$$\mathcal{L}^{\mathsf{SL}}_{\mathsf{pseudo}} := -\log \mathbb{E}_{\tilde{\boldsymbol{y}} \sim p} \sum_{\boldsymbol{y} \models \alpha} \prod_{i=1}^{n} p(\boldsymbol{y}_i \mid \tilde{\boldsymbol{y}}_{-i})$$

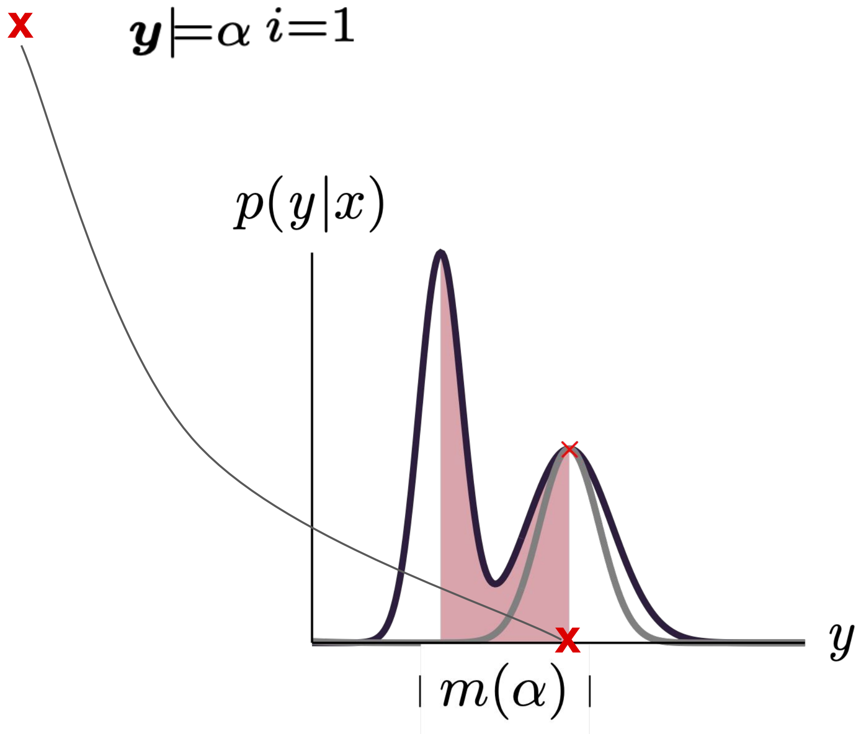
Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

Formally, minimize the *pseudo-semantic loss*

$$\mathcal{L}^{\mathsf{SL}}_{\mathsf{pseudo}} := -\log \mathbb{E}_{\tilde{\boldsymbol{y}} \sim p} \sum_{\boldsymbol{y} \models \alpha} \prod_{i=1}^{n} p(\boldsymbol{y}_i \mid \tilde{\boldsymbol{y}}_{-i})$$

Basic Idea:

Pick a location to build the approximation around

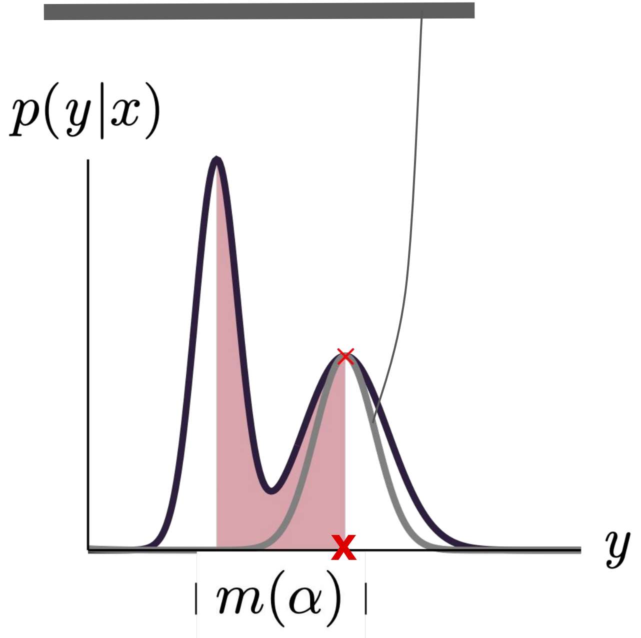


Formally, minimize the *pseudo-semantic loss*

$$\mathcal{L}^{\text{SL}}_{\text{pseudo}} := -\log \mathbb{E}_{\tilde{\boldsymbol{y}} \sim p} \sum_{\boldsymbol{y} \models \alpha} \prod_{i=1}^{n} p(\boldsymbol{y}_i \mid \tilde{\boldsymbol{y}}_{-i})$$

Basic Idea:

Extract a local tractable probabilistic model around the point (independent in each dimension)

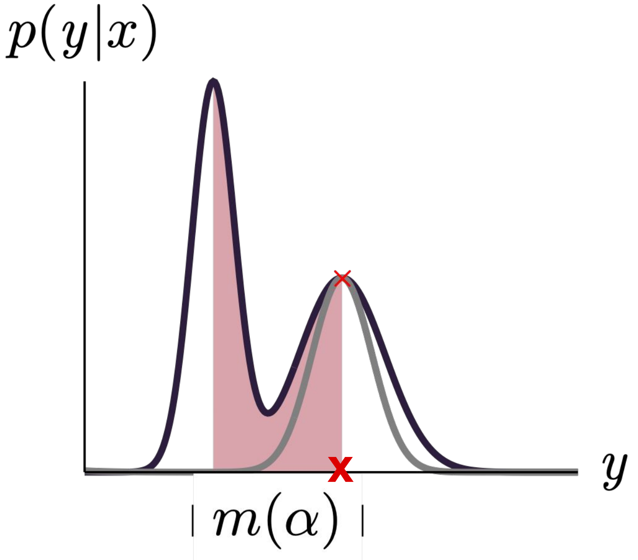


Formally, minimize the *pseudo-semantic loss*

$$\mathcal{L}^{\mathsf{SL}}_{\mathsf{pseudo}} := -\log \mathbb{E}_{\tilde{\boldsymbol{y}} \sim p} \sum_{\boldsymbol{y} \models \alpha} \prod_{i=1}^{n} p(\boldsymbol{y}_i \mid \tilde{\boldsymbol{y}}_{-i})$$

Basic Idea:

Compute Pr(α) locally and maximize it

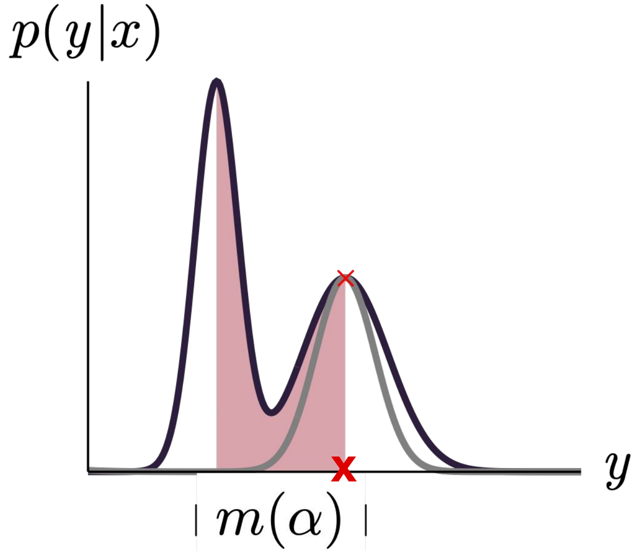


Formally, minimize the *pseudo-semantic loss*

$$\mathcal{L}^{\text{SL}}_{\text{pseudo}} := -\log \mathbb{E}_{\tilde{\boldsymbol{y}} \sim p} \sum_{\boldsymbol{y} \models \alpha} \prod_{i=1}^{n} p(\boldsymbol{y}_i \mid \tilde{\boldsymbol{y}}_{-i})$$

How good is this approximation?

- **Local:**
  ~30 bits entropy vs ~80 for GPT-2.
- **Fidelity:**
  4 bits KL-divergence from GPT-2.



How to compute pseudo-semantic loss?

$$p_{\boldsymbol{\theta}} \sim abc$$

Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

How to compute pseudo-semantic loss?

$$p_{\boldsymbol{\theta}} \sim abc$$

$$\rightarrow \begin{cases} abc & abc & abc \\ \bar{a}bc & a\bar{b}c & ab\bar{c} \end{cases}$$

Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

How to compute pseudo-semantic loss?

$$p_{\boldsymbol{\theta}} \sim abc$$

$$\rightarrow \begin{cases} abc & abc & abc \\ \bar{a}bc & a\bar{b}c & ab\bar{c} \end{cases}$$

$$\rightarrow \begin{cases} p(abc) = 0.13 & p(abc) = 0.13 & p(abc) = 0.13 \\ p(\bar{a}bc) = 0.15 & p(a\bar{b}c) = 0.21 & p(ab\bar{c}) = 0.16 \end{cases}$$

$$\rightarrow \begin{cases} p(a|bc) = 0.46 & p(b|ac) = 0.38 & p(c|ab) = 0.45 \\ p(\bar{a}|bc) = 0.54 & p(\bar{b}|ac) = 0.62 & p(\bar{c}|ab) = 0.55 \end{cases}$$

Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

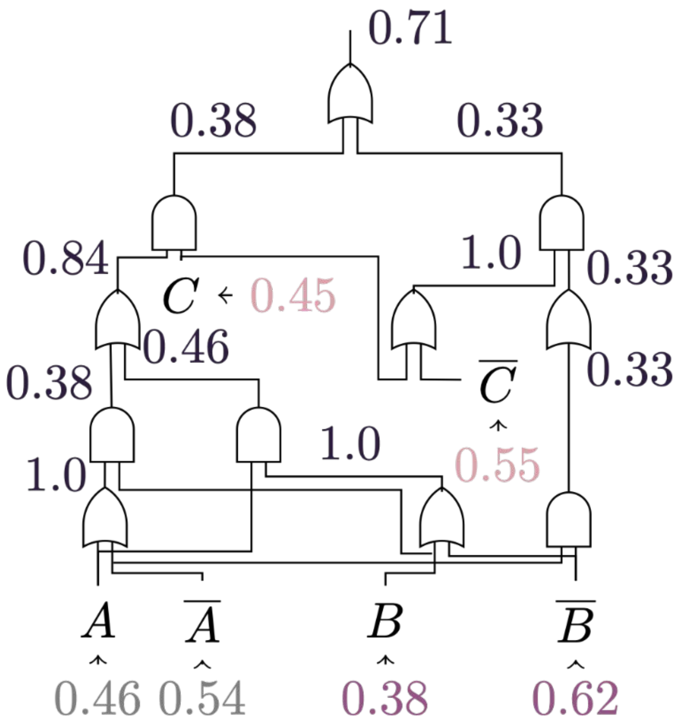
How to compute pseudo-semantic loss?

$$p_{\boldsymbol{\theta}} \sim abc$$

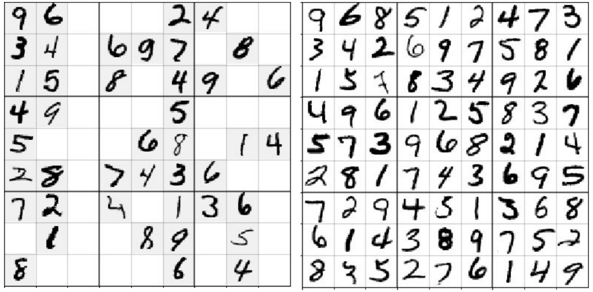
$$\rightarrow \begin{cases} abc & abc & abc \\ \bar{a}bc & a\bar{b}c & ab\bar{c} \end{cases}$$

$$\rightarrow \begin{cases} p(abc) = 0.13 & p(abc) = 0.13 & p(abc) = 0.13 \\ p(\bar{a}bc) = 0.15 & p(a\bar{b}c) = 0.21 & p(ab\bar{c}) = 0.16 \end{cases}$$

$$\rightarrow \begin{cases} p(a|bc) = 0.46 & p(b|ac) = 0.38 & p(c|ab) = 0.45 \\ p(\bar{a}|bc) = 0.54 & p(\bar{b}|ac) = 0.62 & p(\bar{c}|ab) = 0.55 \end{cases}$$



Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.



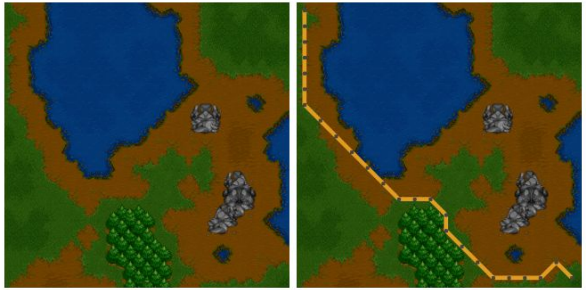


Table 1: Our experimental results on Sudoku.

Test accuracy %	Exact	Consistent
ConvNet	16.80	16.80
ConvNet + SL	22.10	22.10
RNN	22.40	22.40
RNN + PSEUDOSL	**28.20**	**28.20**

Table 2: Our experimental results on Warcraft.

Test accuracy %	Exact	Consistent
ResNet-18	55.00	56.90
ResNet-18 + SL	59.40	61.20
CNN-LSTM	62.00	76.60
CNN-LSTM + PSEUDOSL	**66.00**	**79.00**

Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

Detoxify LLMs by disallowing bad words

Constraint α is a list of 403 toxic words not to say
Evaluation is a toxicity classifier

Models		Exp. Max. Toxicity (↓)			Toxicity Prob. (↓)			PPL (↓)
		Full	Toxic	Nontoxic	Full	Toxic	Nontoxic	
	GPT-2	0.44	0.62	0.39	34.11%	67.27%	24.85%	25.85
Domain-Adaptive	SGEAT [42]	0.32	0.46	0.28	14.05%	35.72%	7.99%	28.72
	PseudoSL *(ours)*	**0.29**	0.38	**0.27**	9.80%	20.07%	6.93%	28.14
Word Banning	GPT-2	0.40	0.55	0.36	27.92%	57.86%	19.56%	22.24
	SGEAT [42]	0.30	0.41	**0.27**	10.73%	27.05%	**6.17%**	24.91
	PseudoSL *(ours)*	**0.29**	**0.37**	**0.27**	**9.20%**	**18.71%**	6.55%	24.19

Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

Outline

1. The paradox of learning to reason from data

~~end-to-end learning~~

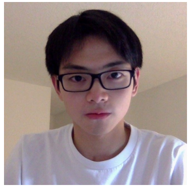
2. Symbolic reasoning at generation time

3. Symbolic reasoning at training time

logical + *probabilistic reasoning* + *deep learning*

Thanks

This was the work of many wonderful students/postdocs/collaborators!



Honghua

Kareem

Meihua

References: http://starai.cs.ucla.edu/publications/