

UCLA

**Computer
Science**



Symbolic Reasoning for Large Language Models

Guy Van den Broeck

AIxIA-AISB Spotlight Seminar - Jun 20 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.

<p>Kristin and her son Justin went to visit her mother Carol on a nice Sunday afternoon. They went out for a movie together and had a good time.</p> 	<p>Q: How is Carol related to Justin ?</p> <p>A: Carol is the grandmother of Justin</p> 
--	---

Reasoning Example
from the CLUTRR
dataset

Unclear whether they follow the rules of logical deduction.

Language Models:

input → ? → *Carol is the grandmother of Justin.*

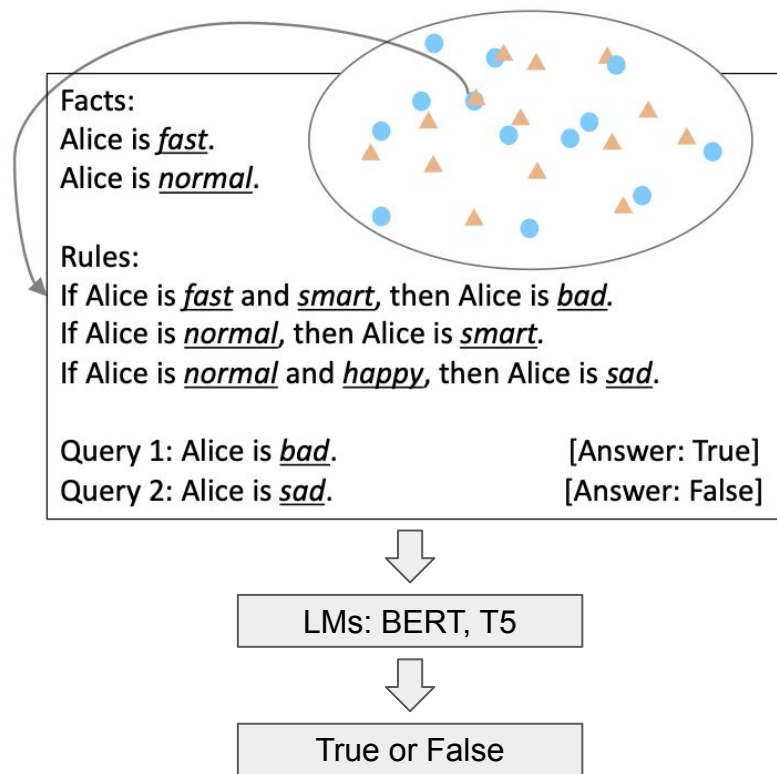
Logical Reasoning:

input → *Justin is 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 ($\approx 10^{360}$)
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



SimpleLogic

Generate textual train and test examples of the form:

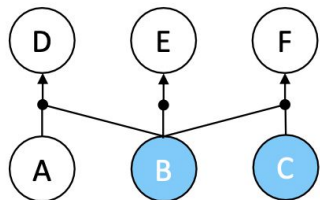
Rules: If witty, then diplomatic. If careless and condemned and attractive, then blushing. If dishonest and inquisitive and average, then shy. If average, then stormy. If popular, then blushing. If talented, then hurt. If popular and attractive, then thoughtless. If blushing and shy and stormy, then 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 ?

Training a transformer on SimpleLogic

(1) Randomly sample facts & rules.

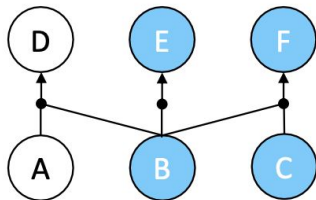
Facts: B, C

Rules: $A, B \rightarrow D$. $B \rightarrow E$. $B, C \rightarrow F$.



Rule-Priority

(2) Compute the correct labels for all predicates given the facts and rules.



Label-Priority



(1) Randomly assign labels to predicates.

True: B, C, E, F.

False: A, D.

(2) Set B, C (randomly chosen among B, C, E, F) as facts and sample rules (randomly) consistent with the label assignments.

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

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!



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.

Why? Statistical Features

Monotonicity of entailment:

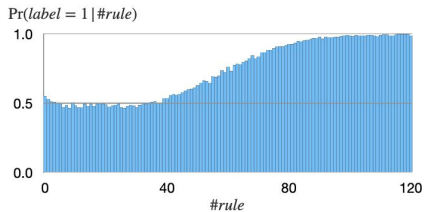
Any rules can be freely added to the axioms of any proven fact.



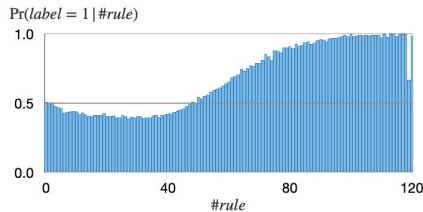
The more rules given, the more likely a predicate will be proven.



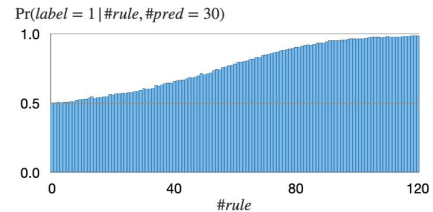
$\Pr(\text{label} = \text{True} \mid \text{Rule \#} = x)$ should increase (roughly) monotonically with x



(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 $\Pr(\text{label} = \text{True} \mid \text{rule\#} = x) = 0.5$ 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”

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...

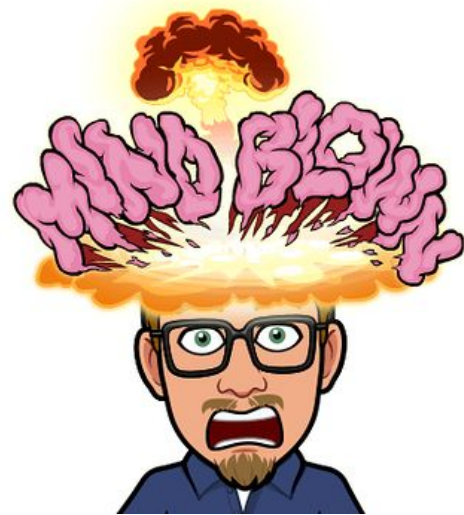
FAQ: Will reasoning emerge? Perhaps on 99% of human behavior...

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.

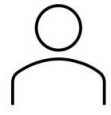
Initialize the model with the perfect parameters that simulate the ground-truth reasoning algorithm.

Then SGD will **un-learn the algorithm** that generalizes OOD
... we don't understand what is going on ...

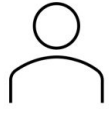


Outline

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.

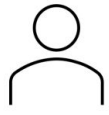


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



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

ChatGPT

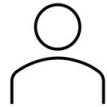


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



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

ChatGPT

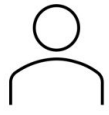


That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.



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

ChatGPT

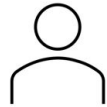


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



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

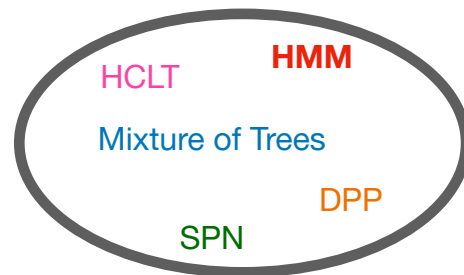
Tractable Probabilistic Models (TPMs)
model **joint probability distributions**
and allow **efficient** probabilistic inference.

e.g., efficient marginalization:

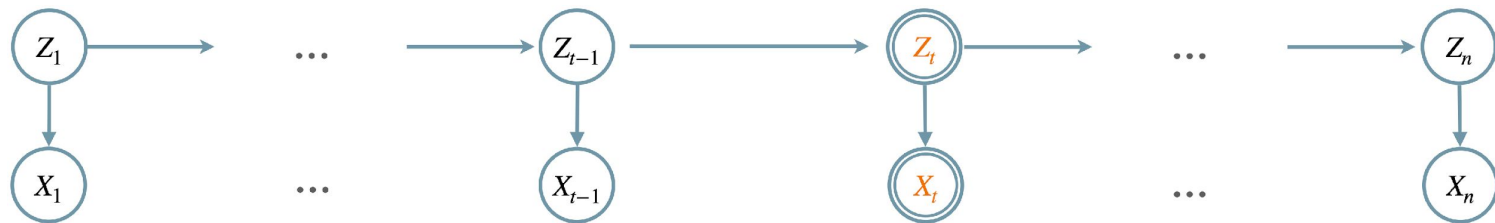
$$p_{\text{TPM}}(\text{3rd token} = \text{frisbee}, \text{5th token} = \text{dog})$$

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

Probabilistic Circuits



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 $\text{KL}(p_{\text{gpt}} // p_{\text{HMM}})$
3. Leverages latent variable distillation for training at scale [ICLR 23].
(Cluster embeddings of examples to estimate latent Z_i)

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

For constraint α in CNF:

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

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

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

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

Each clause represents the inflections for one keyword.

GeLaTo Overview



Lexical Constraint α : sentence contains keyword "winter"

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

✗ intractable

✓ efficient

Pre-trained
Language Model

Tractable
Probabilistic Model

Minimize KL-divergence

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

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

GeLaTo Overview



Lexical Constraint α : sentence contains keyword "winter"

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

✗ intractable

✓ efficient

Pre-trained
Language Model

Tractable
Probabilistic Model

Minimize KL-divergence

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

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

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

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

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

$$p_{\cancel{lm}}(\text{next-token} \mid \text{prefix}, \alpha) \propto p_{\cancel{lm}}(\alpha \mid \text{next-token}, \text{prefix}) \cdot p_{lm}(\text{next-token} \mid \text{prefix})$$

gelato *hmm*



Method	Generation Quality								Constraint Satisfaction			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>
<i>Unsupervised</i>												
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

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

Supervised

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

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

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

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

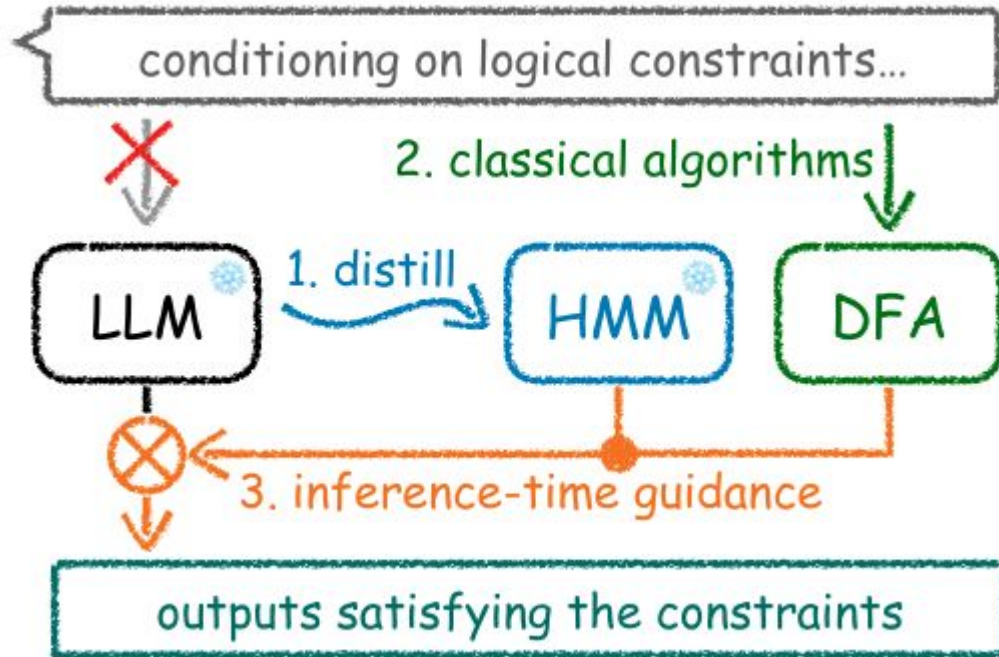
Method	Generation Quality								Constraint Satisfaction			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>
<i>Supervised</i>												
NeuroLogic (Lu et 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

Advantages of GeLaTo:

1. Constraint α is guaranteed to be satisfied:
for any next-token x_{t+1} that would make α unsatisfiable, $p(x_{t+1} | x_{1:t}, \alpha) = 0$.
2. Training p_{hmm} does not depend on α ,
which is only imposed at inference (generation) time.
3. Can impose additional tractable constraints:
 - 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.

What's next? Tractable Control with Ctrl-G



What's next? Tractable Control with Ctrl-G

User: given the following context, generate infilling text for [BLANK] using key phrases "alien mothership", "far from over"; generated text must contain 25 - 30 words.

"First they've defeated a small squad [BLANK] are few humans left, and despite their magical power, their numbers are getting fewer."

5 lines of code!

```
from CtrlG import *  
  
prefix = "First they defeated a ..."  
suffix = "are few humans left ..."  
  
dfa_list = [  
    DFA_all_of("alien mothership",  
              "far from over"),  
    DFA_word_count(25, 30),  
]  
dfa = DFA_logical_and(dfa_list)  
  
lp = CtrlGLogitsProcessor(  
    dfa, hmm, prefix, suffix)  
llm.generate(logits_processor=lp)
```

"First they've defeated a small squad of aliens, then a larger fleet of their ships. Eventually they've even managed to take down the alien mothership. But their problems are far from over. There are few humans left, and despite their magical power, their numbers are getting fewer."

What's next?

Tractable Control with Ctrl-G

	<i>Insertion</i>			
	<i>None</i>	<i>K</i>	<i>L</i>	<i>K&L</i>
<i>Quality</i>				
TULU2	2.68	2.64	2.78	2.74
GPT3.5	2.27	2.22	2.27	2.31
GPT4	3.79	3.33	3.53	3.10
Ctrl-G	3.77	3.56	3.73	3.59
<i>Success</i>				
TULU2	-	12%	20%	3%
GPT3.5	-	22%	54%	10%
GPT4	-	60%	20%	27%
Ctrl-G	-	100%	100%	100%
<i>Overall</i>				
TULU2	-	7%	10%	1%
GPT3.5	-	0%	5%	2%
GPT4	-	41%	17%	14%
Ctrl-G	-	76%	78%	82%

Table 3: Human evaluation of interactive text editing. *K&L* indicates that the model should adhere to both keyphrase (*K*) and word length (*L*) constraints simultaneously. We present the human evaluation score (*Quality*), constraint success rate (*Success*), and overall satisfaction rate (*Overall*), which represents the proportion of examples meeting logical constraints with a *Quality* score above 3.

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 $\text{Pr}(\cdot)$ that

1. satisfies the constraint α : $\text{Pr}(\alpha) \uparrow$
2. maximizes the likelihood: $\text{Pr}(D) \uparrow$

Neurosymbolic learning of transformers

Given:

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

Learn: a transformer $\text{Pr}(\cdot)$ that

1. satisfies the constraint α : $\text{Pr}(\alpha) \uparrow$
2. maximizes the likelihood: $\text{Pr}(D) \uparrow$

$\text{Pr}(\alpha)$ is **computationally hard**, even when α is trivial:

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

Autoregressive distributions are hard...

$\Pr(\alpha)$ is **computationally hard**, even when α is trivial:

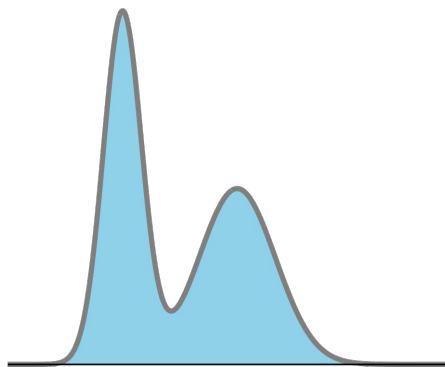
What is prob. that LLM ends the sentence with “UCLA”?

Why did it work before?

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

Now we need to train the actual intractable transformer...

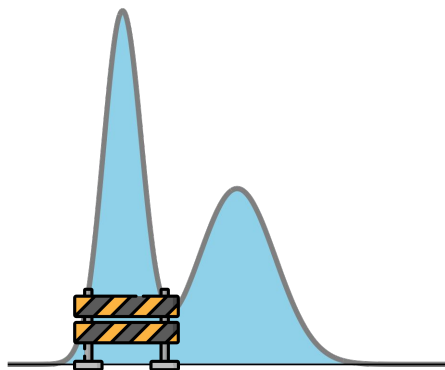
Neuro-Symbolic AI: A Probabilistic Perspective



A neural network
induces a distribution

Neuro-Symbolic AI: A Probabilistic Perspective

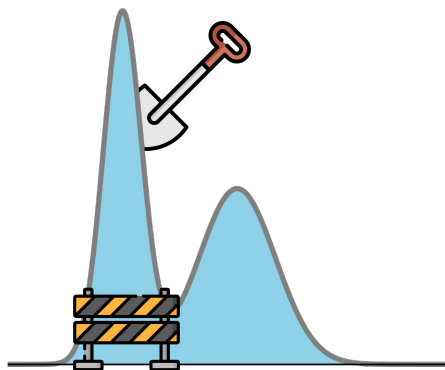
Impose structure
using symbolic
knowledge



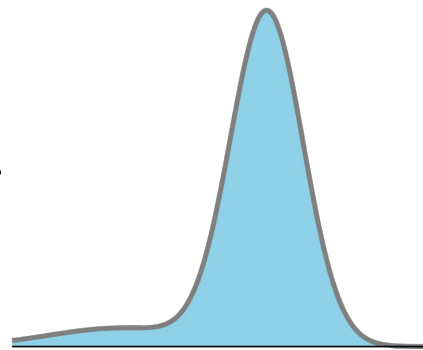
A neural network
induces a distribution

Neuro-Symbolic AI: A Probabilistic Perspective

Impose structure
using symbolic
knowledge



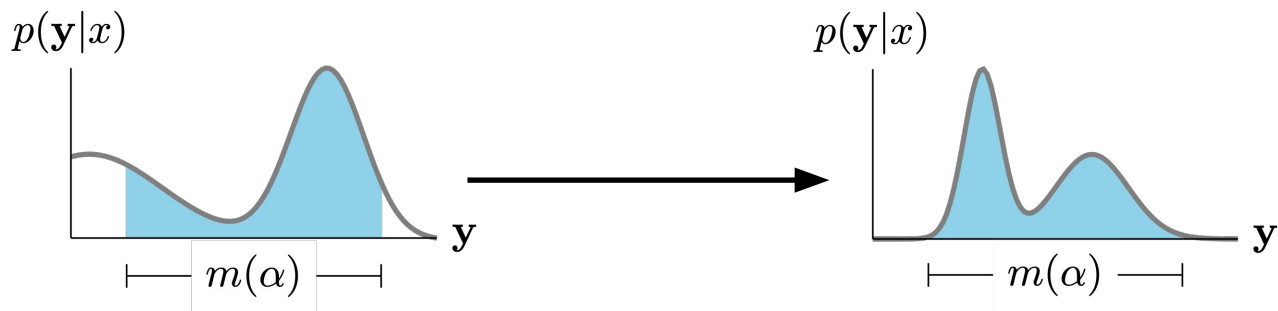
A neural network
induces a distribution



Move mass around to be
consistent with structure

The Problem

We want to shift the model's output distribution away from violating the constraint



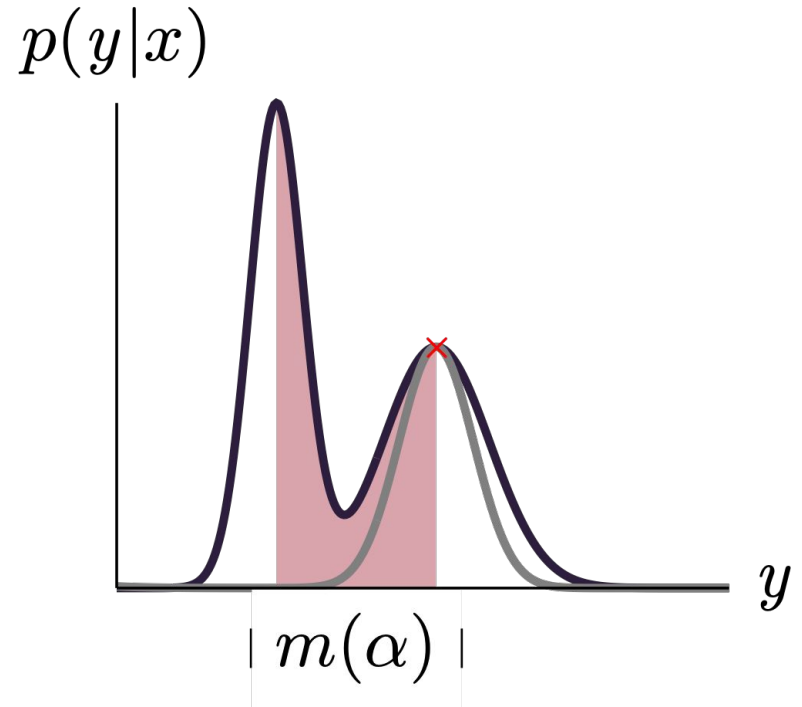
Easy when p is fully-independent; *very hard* when p is autoregressive

$$p(\mathbf{y}) = \prod_{i=1}^n p(y_i | y_{<i}),$$

Neurosymbolic learning of transformers

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.



Formally, minimize the *pseudo-semantic loss*

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

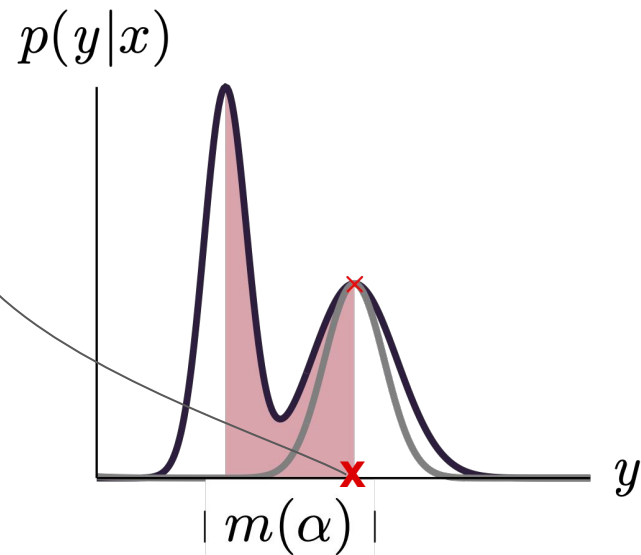


Formally, minimize the *pseudo-semantic loss*

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

Basic Idea:

Pick a location to build the approximation around

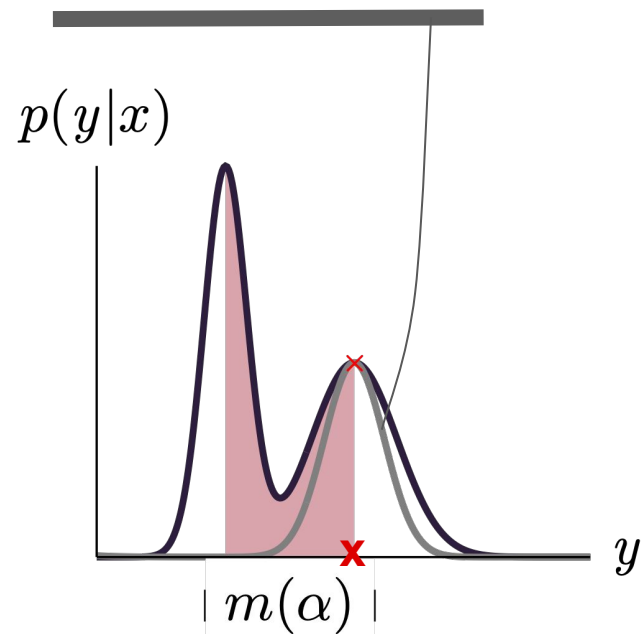


Formally, minimize the *pseudo-semantic loss*

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

Basic Idea:

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



How to compute pseudo-semantic loss?

$\tilde{y} :=$ I saw a dog today

$$p(\text{I saw a dog today}) = p(\text{I}) \times p(\text{saw}|\text{I}) \times p(\text{a}|\text{I, saw}) \times p(\text{dog}|\text{I, saw, a}) \times p(\text{today}|\text{I, saw, a, dog})$$

$p(\text{She}) \quad p(\text{caught}|\text{I}) \quad p(\text{the}|\text{I, saw}) \quad p(\text{cat}|\text{I, saw, a}) \quad p(\text{yesterday}|\text{I, saw, a, dog})$
 $p(\text{He}) \quad p(\text{bought}|\text{I}) \quad p(\text{an}|\text{I, saw}) \quad p(\text{mouse}|\text{I, saw, a}) \quad p(\text{tomorrow}|\text{I, saw, a, dog})$
 $\dots \quad \dots \quad \dots \quad \dots \quad \dots$

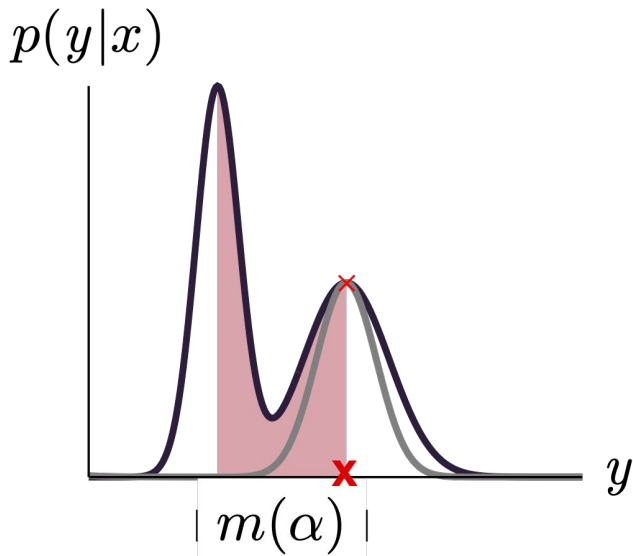
$$p(\text{I saw a mouse today}) = p(\text{I}) \times p(\text{saw}|\text{I}) \times p(\text{a}|\text{I, saw}) \times p(\text{mouse}|\text{I, saw, a}) \times p(\text{today}|\text{I, saw, a, dog})$$

Formally, minimize the *pseudo-semantic loss*

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

Basic Idea:

Compute $\Pr(\alpha)$ locally and maximize it

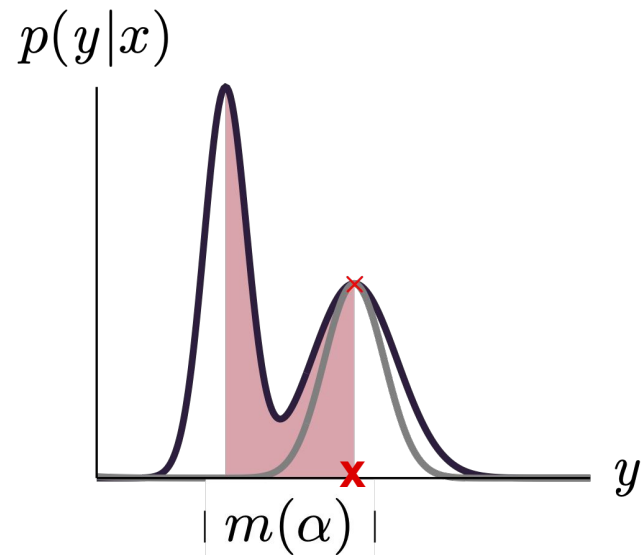


Formally, minimize the *pseudo-semantic loss*

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

How good is this approximation?

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



9	6			2	4			9	6	8	5	1	2	4	7	3	
3	4		6	9	7		8	3	4	2	6	9	7	5	8	1	
1	5		8	4	9			1	5	7	8	3	4	9	2	6	
4	9				5			4	9	6	1	2	5	8	3	7	
5				6	8		1	4	5	7	3	9	6	8	2	1	4
2	8		7	4	3	6			2	8	1	7	4	3	6	9	5
7	2		4		1	3	6		7	2	9	4	5	1	3	6	8
	1			8	9		5		6	1	4	3	8	9	7	5	2
8					6		4		8	3	5	2	7	6	1	4	9

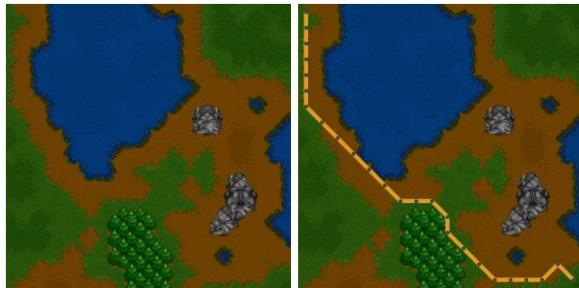


Table 1: Our experimental results on Sudoku.

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

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 (\downarrow)			Toxicity Prob. (\downarrow)			PPL (\downarrow)	
	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 (<i>ours</i>)	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 (<i>ours</i>)	0.29	0.37	0.27	9.20%	18.71%	6.55%	24.19

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/>