UCLA Computer Science

STAR AI
RESEARCH LAB
UCLA

(Probabilistic Circuits for)

Neurosymbolic Reasoning for Large Language Models

Guy Van den Broeck

Neuro-Symbolic AI Summer School - Sep 5 2024

Outline

1. A neurosymbolic problem hidden in LLMs
2. The paradox of learning to reason from data

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

3. Symbolic reasoning at generation time

4. Symbolic reasoning at training time

logical + probabilistic reasoning + deep learning

Outline

1. **A neurosymbolic problem hidden in LLMs**
2. The paradox of learning to reason from data

~~end-to-end learning~~

3. Symbolic reasoning at generation time

4. Symbolic reasoning at training time

logical + probabilistic reasoning + deep learning

Tokenization in NLP

Most language models represent distributions over *tokens* (subwords), not strings.

$$\textbf{string} \quad \mathbf{x} = (x_1, x_2, \ldots, x_n)$$
$$\textbf{tokenization} \quad \mathbf{v} = (v_1, \ldots, v_m)$$

For example:

$$\textbf{string} \quad \mathbf{x} = \texttt{Caterpillar}$$
$$\textbf{tokenization} \quad \mathbf{v} = [\texttt{C},\texttt{ater},\texttt{p},\texttt{ill},\texttt{ar}] \equiv [315,1008,29886,453,279]$$

↪ Why *tokens* instead of *bytes*?

$p_{\mathrm{LLM}}([\texttt{C},\texttt{a},\texttt{t},\texttt{e},\texttt{r},\texttt{p},\texttt{i},\texttt{l},\texttt{l},\texttt{a},\texttt{r}])$ requires $|\mathbf{x}|$ calls to the LLM (for autoregressive models, e.g. transformers)

Harder to capture long term dependency

↪ Why *tokens* instead of *words*?

Robustness to typos and new words

Tokens capture morphology

Tokenization in NLP

How do you do learning and inference?

Define a unique *canonical* tokenization of a string!

Example:

$$\textbf{string} \quad \mathbf{x} = \texttt{Caterpillar}$$
$$\textbf{canonical} \quad \mathbf{v} = [\texttt{C},\texttt{ater},\texttt{p},\texttt{ill},\texttt{ar}]$$

Common assumption:

$$p(\mathbf{x}) = p(\mathbf{v}) \qquad ✗$$

A string can be tokenized in an exponential number of ways (784 here!)

$[\texttt{C},\texttt{ater},\texttt{pi},\texttt{l},\texttt{lar}]$, $[\texttt{Cat},\texttt{er},\texttt{pi},\texttt{lla},\texttt{r}]$, $[\texttt{Cat},\texttt{er},\texttt{pi},\texttt{l},\texttt{lar}]$,
$[\texttt{Ca},\texttt{ter},\texttt{p},\texttt{ill},\texttt{ar}]$, $[\texttt{Ca},\texttt{ter},\texttt{p},\texttt{illa},\texttt{r}]$, $[\texttt{Cat},\texttt{er},\texttt{pi},\texttt{ll},\texttt{ar}]$,
$\ldots$
$[\texttt{Ca},\texttt{t},\texttt{e},\texttt{r},\texttt{p},\texttt{i},\texttt{l},\texttt{l},\texttt{a},\texttt{r}]$, $[\texttt{C},\texttt{a},\texttt{t},\texttt{e},\texttt{r},\texttt{p},\texttt{i},\texttt{l},\texttt{l},\texttt{a},\texttt{r}]$

Tokenization in NLP

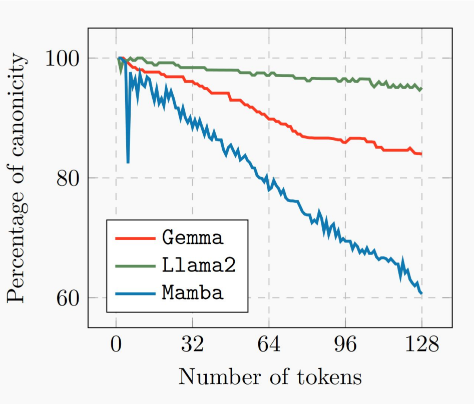
Why does this tokenization problem matter?

$$\text{string} \quad \mathbf{x} = \texttt{Hypnopaturist}$$
$$\text{canonical} \quad \mathbf{v} = [\texttt{Hyp,nop,atu,rist}]$$
$$\text{most likely} \quad \mathbf{v} = [\texttt{Hyp,no,patu,rist}]$$

$$\text{canonical prob} \quad p(\mathbf{v}|\mathbf{x}) \approx 0.0004$$
$$\text{most likely prob} \quad p(\mathbf{v}|\mathbf{x}) \approx 0.9948$$

Less likely for non-English (code, unicode characters, etc)

How canonical are unconditional samples?



We're ignoring an exponential number of tokenizations!

Tokenization is a Neurosymbolic Problem!

But why do we care? What is the neurosymbolic problem here?

- ↪ Tokens are symbols.
- ↪ A tokenization of a text is a constraint over these symbols.

$$p(\mathbf{v}, \mathbf{x}) = \begin{cases} p_{\mathrm{LLM}}(\mathbf{v}) & \text{if } \mathbf{v} \models \mathbf{x}; \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbf{v} = (v_1, v_2, \ldots, v_n) \models \mathbf{x} \Leftrightarrow v_1 \circ v_2 \circ \cdots \circ v_n = \mathbf{x}$$

concatenation

Example:

$$p(\mathbf{v} = [\text{一 ラ}, \text{ク}] \,|\, \mathbf{x} = \text{一ラク}) = 0.586$$

$$p(\mathbf{v} = [\text{一}, \text{ラ}, \text{ク}] \,|\, \mathbf{x} = \text{一ラク}) = 0.012$$

$$p(\mathbf{v} = [\text{一}, \text{ラク}] \,|\, \mathbf{x} = \text{一ラク}) = 0.402$$

$$p(\mathbf{v} = [\texttt{Tok}, \texttt{ens}] \,|\, \mathbf{x} = \text{一ラク}) = 0$$

From Gemma 2B

Tokenization in NLP

How do you do learning and inference?

Define a unique *canonical* tokenization of a string!

Example:

$$\text{string} \quad \mathbf{x} = \texttt{Caterpillar}$$
$$\text{canonical} \quad \mathbf{v} = [\texttt{C,ater,p,ill,ar}]$$

Common assumption:

$$p(\mathbf{x}) = p(\mathbf{v}) \quad ✗ \qquad p(\mathbf{x}) = \sum_{\mathbf{v} \models \mathbf{x}} p(\mathbf{v}) \quad ✔$$

A string can be tokenized in an exponential number of ways (784 here!)

$[\texttt{C,ater,pi,l,lar}]$, $[\texttt{Cat,er,pi,lla,r}]$, $[\texttt{Cat,er,pi,l,lar}]$,
$[\texttt{Ca,ter,p,ill,ar}]$, $[\texttt{Ca,ter,p,illa,r}]$, $[\texttt{Cat,er,pi,ll,ar}]$,
$\ldots$
$[\texttt{Ca,t,e,r,p,i,l,l,a,r}]$, $[\texttt{C,a,t,e,r,p,i,l,l,a,r}]$

Reasoning in Tokenization Space

Instead of the *canonical* tokenization, we might want to compute:

1. The most likely tokenization

$$\arg\max_{\mathbf{v} \models \mathbf{x}} p(\mathbf{v}, \mathbf{x})$$

✘

Theorem. *The most likely tokenization problem is NP-hard.*

For autoregressive models, e.g. transformers and state space models

2. The true marginal probability of a text

$$p(\mathbf{x}) = \sum_{\mathbf{v} \models \mathbf{x}} p(\mathbf{v}, \mathbf{x})$$

✘

Theorem. *The marginal string probability problem is #P-hard.*

Proof Intuition: Choice of tokens can encode Boolean variables,
LLM probability encodes which clauses in a CNF are satisfied

Renato Lui Geh, Honghua Zhang, Kareem Ahmed, Benjie Wang and Guy Van den Broeck. Where is the signal in tokenization space?, 2024

(Approximate) Reasoning in Tokenization Space

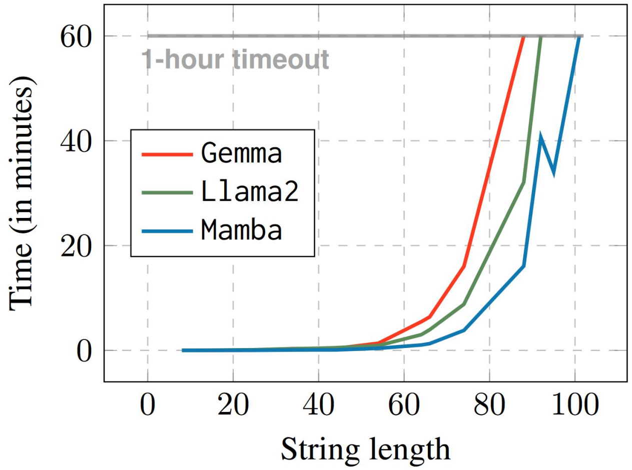
1. The most likely tokenization

$$\arg\max_{\mathbf{v} \models \mathbf{x}} p(\mathbf{v}, \mathbf{x})$$

Branch-and-bound

- ↪ Lower bound: canonical likelihood
- ↪ Anytime: candidate at least as good as canonical
- ↪ Runtime exponential on string length!
- ↪ Canonical best candidate for almost all cases…

…but not always!



whitespace character

$p(\mathbf{v} = [\texttt{␣tongue,less}] \mid \mathbf{x} = \texttt{␣tongueless}) = 0.518$ → most likely tokenization

$p(\mathbf{v} = [\texttt{␣t,ong,uel,ess}] \mid \mathbf{x} = \texttt{␣tongueless}) = 0.004$

$p(\mathbf{v} = [\texttt{␣tong,uel,ess}] \mid \mathbf{x} = \texttt{␣tongueless}) = 0.474$

canonical tokenization

$p(\mathbf{v} = [\texttt{␣,HEADER,\_,DELIM,ITER}] \mid \mathbf{x} = \texttt{␣HEADER\_DELIMITER}) = 0.412$

$p(\mathbf{v} = [\texttt{␣HEAD,ER,\_,DELIM,ITER}] \mid \mathbf{x} = \texttt{␣HEADER\_DELIMITER}) = 0.330$

$p(\mathbf{v} = [\texttt{␣HEADER,\_,DELIM,ITER}] \mid \mathbf{x} = \texttt{␣HEADER\_DELIMITER}) = 0.010$

canonical tokenization

Is there signal in non-canonical tokenizations?

From Gemma 2B

Renato Lui Geh, Honghua Zhang, Kareem Ahmed, Benjie Wang and Guy Van den Broeck. Where is the signal in tokenization space?, 2024

(Approximate) Reasoning in Tokenization Space

2. The true probability of a text

$$p(\mathbf{x}) = \sum_{\mathbf{v} \models \mathbf{x}} p(\mathbf{v}, \mathbf{x})$$

Sequential importance sampling

$$p(\mathbf{x}) = \mathbb{E}_{\mathbf{v} \sim q(\mathbf{v}|\mathbf{x})} \left[\frac{p(\mathbf{v}, \mathbf{x})}{q(\mathbf{v}|\mathbf{x})} \right]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \frac{p\left(\mathbf{v}^{(i)}, \mathbf{x}\right)}{q\left(\mathbf{v}^{(i)} | \mathbf{x}\right)}$$

LLM forward pass

proposal distribution

Unbiased estimator converging to the true probability of text as #samples grows

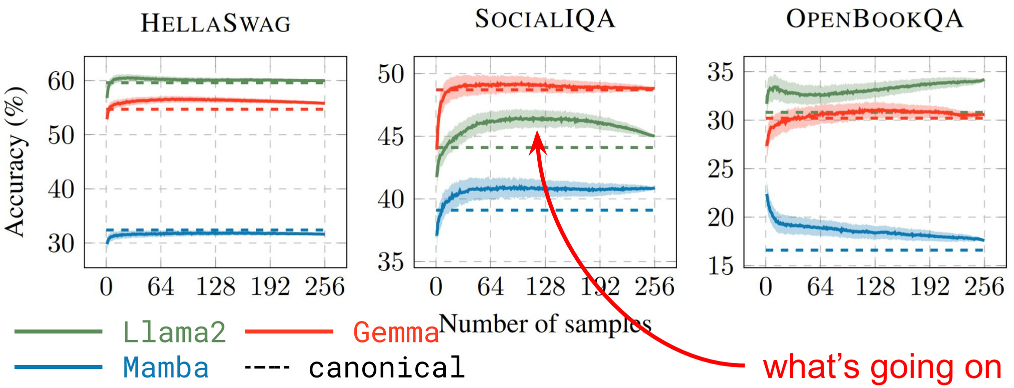
One step look-ahead proposal distribution:

$$q_{\mathrm{LA}}(v_j | \mathbf{v}_{1:j-1} = [\texttt{Tok,eni}], \mathbf{x} = \texttt{Tokenization}) = \begin{cases} 0.50 & \text{, if } v_j = \texttt{zation}; \\ 0.30 & \text{, if } v_j = \texttt{zat}; \\ 0.15 & \text{, if } v_j = \texttt{za}; \\ 0.05 & \text{, if } v_j = \texttt{z}; \\ 0.00 & \text{, if } v_j = \texttt{a}; \\ \vdots & \\ 0.00 & \text{, if } v_j = \texttt{zzz}; \end{cases}$$

zero-out next tokens inconsistent with constraint

Renato Lui Geh, Honghua Zhang, Kareem Ahmed, Benjie Wang and Guy Van den Broeck. Where is the signal in tokenization space?, 2024

Where is the signal in tokenization space?



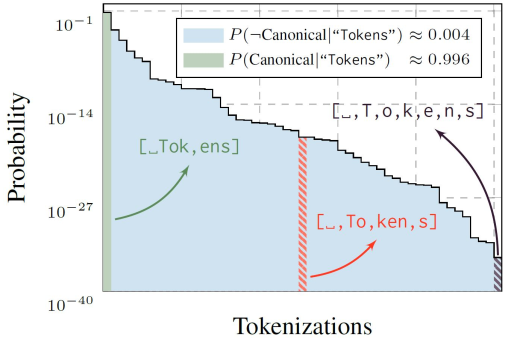
$$\arg\max_{\text{answer}} p(\text{answer}|\mathbf{v}_{\text{question}}) = \sum_{\mathbf{v}\models\text{answer}} p(\mathbf{v}|\mathbf{v}_{\text{question}})$$

what's going on here?

California experiences heavy earthquake activity due to

(a) erosion
(b) techtonics
(c) volcanic activity
(d) fire

Most of the time, canonical is overwhelmingly more likely in English.



So text probability estimate will eventually converge to canonical in almost all cases.

But before it does, non-canonical tokenizations are given more weight!

There is signal in non-canonical tokenizations!

Renato Lui Geh, Honghua Zhang, Kareem Ahmed, Benjie Wang and Guy Van den Broeck. Where is the signal in tokenization space?, 2024

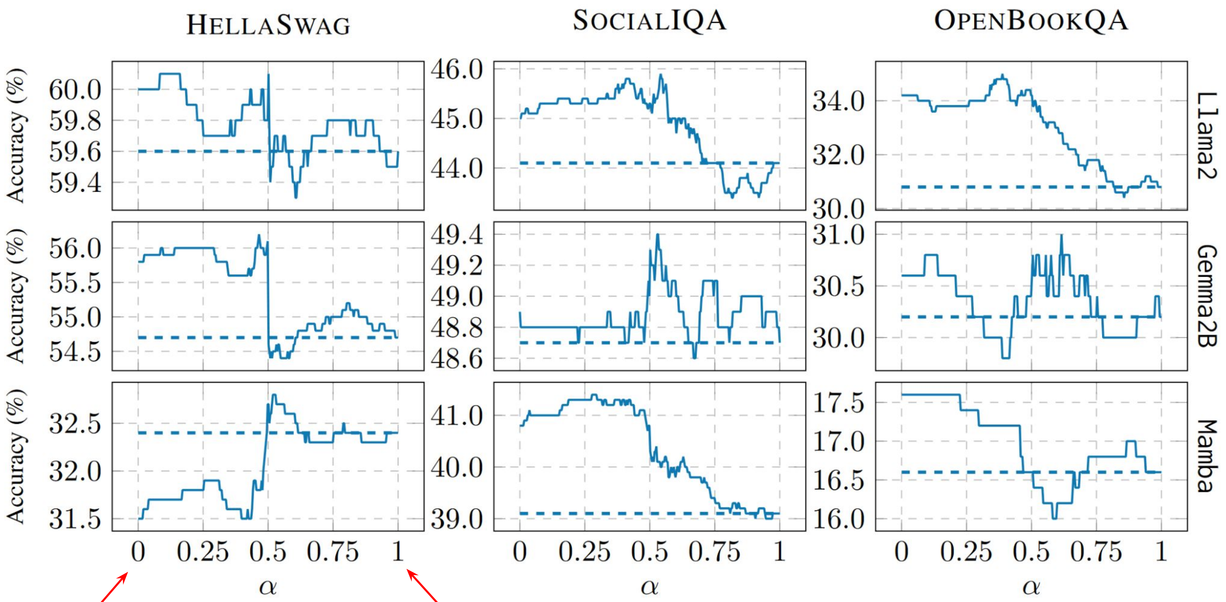
Neurosymbolic reasoning can boost LLM accuracy!

Can we quantify how much signal is in non-canonical tokenizations?

$$\arg\max_{\text{answer}} \alpha \cdot \underline{p(\mathbf{v}_{\text{answer}} | \mathbf{v}_{\text{question}})} + (1-\alpha) \cdot \underline{p(\text{noncanonical} | \mathbf{v}_{\text{question}})}$$

canonical

non-canonicals



only non-canonicals

only canonical

Tune for α

	Mixture	Canonical	
	Accuracy (%)		
Llama2	**59.7**	59.6	HellaSwag
Gemma	**55.8**	54.7	
Mamba	31.6	**32.4**	
Llama2	**44.8**	44.1	SocialIQA
Gemma	**48.8**	48.7	
Mamba	**39.8**	39.1	
Llama2	**34.0**	30.8	OpenBookQA
Gemma	**30.6**	30.2	
Mamba	**17.6**	16.6	

Consistent improvement!

Renato Lui Geh, Honghua Zhang, Kareem Ahmed, Benjie Wang and Guy Van den Broeck. Where is the signal in tokenization space?, 2024

Outline

1. A neurosymbolic problem hidden in LLMs
2. **The paradox of learning to reason from data**

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

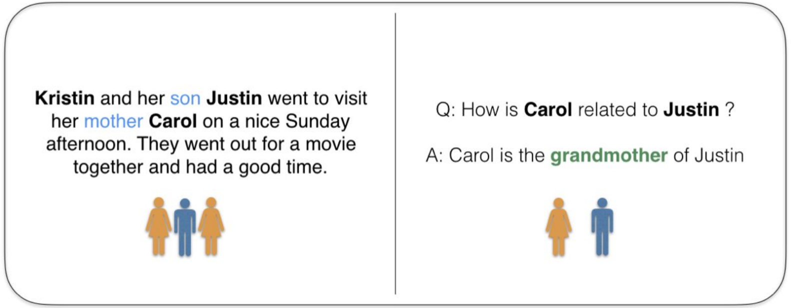
3. Symbolic reasoning at generation time

4. 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.

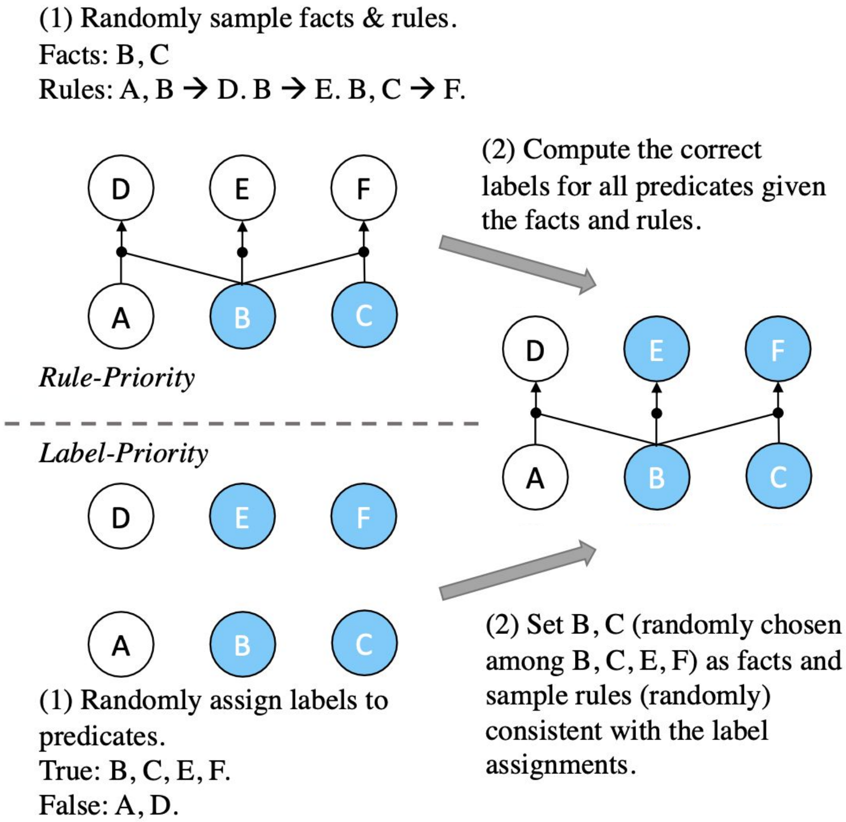
SimpleLogic

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.

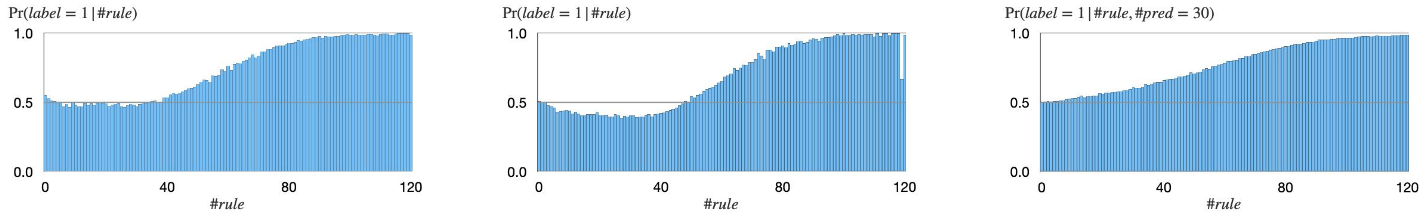
Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, 2022

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

Outline

1. A neurosymbolic problem hidden in LLMs
2. The paradox of learning to reason from data

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

3. **Symbolic reasoning at generation time**

4. 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 $q(\text{next-token}|\text{prefix}, \alpha)$ and avoid symbolic reasoning

What do we need?

Prefix: "The weather is"

Constraint α: text contains "winter"

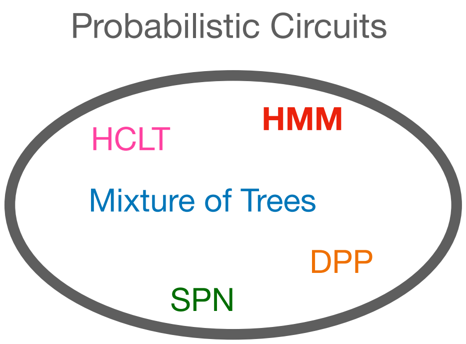
Generate from $p(\text{next-token}|\text{prefix}, \alpha) =$

cold	0.50
warm	0.01

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

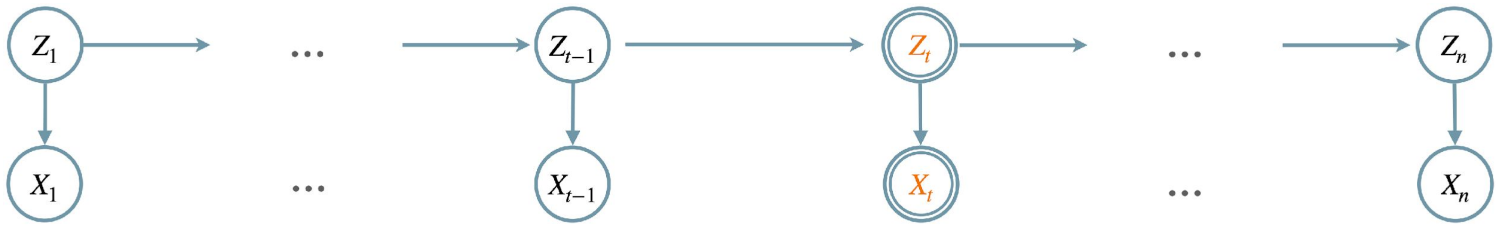
Marginalization!

Tractable Probabilistic Models



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

For now… keep it simple… just a Hidden Markov Model (HMM) with 4096 hidden states and 50k emission tokens



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

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

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.

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

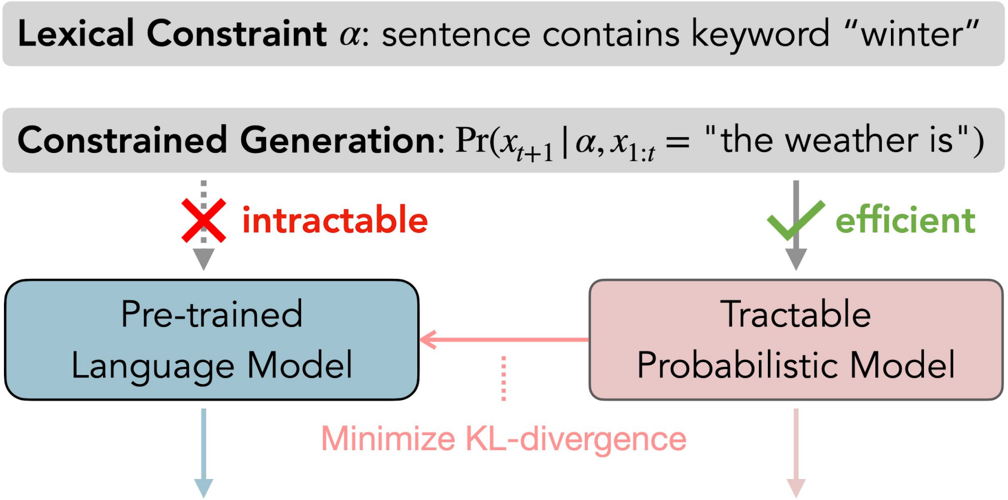
Reference 1: A car drives down a snow covered road.

Reference 2: Two cars drove through the snow.

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

Each clause represents the inflections for one keyword.

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

efficient

Pre-trained Language Model

Tractable Probabilistic 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

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

gelato hmm

GPT2-large (774M params) + HMM (212M params)

Method	*Generation Quality* ROUGE-L		BLEU-4		CIDEr		SPICE		*Constraint Satisfaction* 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.

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.

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

What about more powerful constraints?
more powerful LLMs?

More powerful constraints? 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."

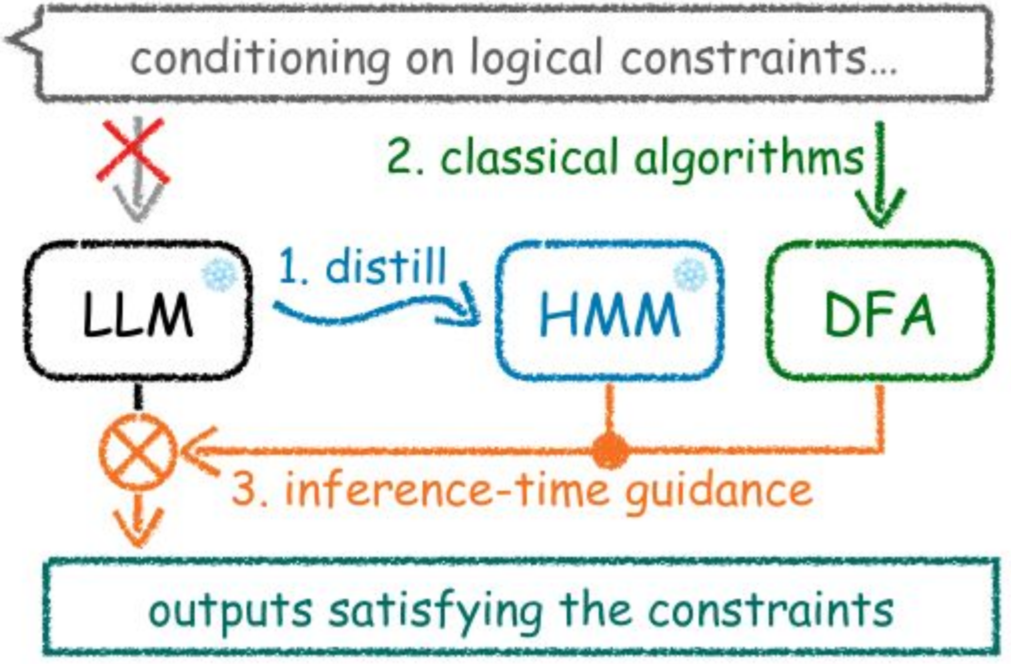
5 lines of code!

```
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."

Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck and Nanyun Peng. Adaptable Logical Control for Large Language Models, *In Arxiv*, 2024.

More powerful constraints?
Tractable Control with Ctrl-G



Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck and Nanyun Peng. Adaptable Logical Control for Large Language Models, *In Arxiv*, 2024.

Tractable Control with Ctrl-G

Ctrl-G (applied to TULU2-7B) significantly outperforms GPT4 in generating text continuations/insertions under constraints. Notably for *insertion*, while GPTs produce lower quality outputs as the constraints become more complex, Ctrl-G **consistently** produce high-quality output.

	Insertion			
	None	*K*	*L*	*K&L*
Quality				
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**
Success				
TULU2	-	12%	20%	3%
GPT3.5	-	22%	54%	10%
GPT4	-	60%	20%	27%
Ctrl-G	-	**100%**	**100%**	**100%**
Overall				
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.

Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck and Nanyun Peng. Adaptable Logical Control for Large Language Models, *In Arxiv*, 2024.

Outline

1. A neurosymbolic problem hidden in LLMs
2. The paradox of learning to reason from data

~~end-to-end learning~~

3. Symbolic reasoning at generation time

4. **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 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)↑

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 “UCLA”?

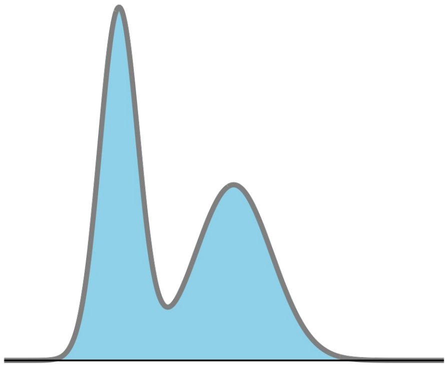
Why did it work before?

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.

Neuro-Symbolic AI: A Probabilistic Perspective

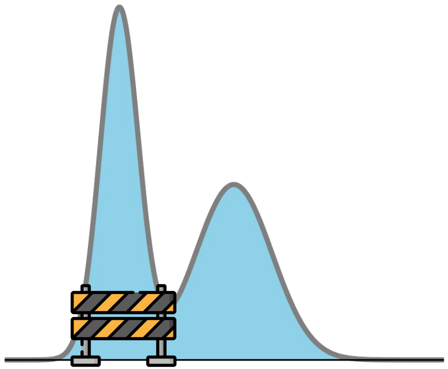


A neural network
induces a distribution

[Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang and Guy Van den Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge, *ICML*, 2018]

Neuro-Symbolic AI: A Probabilistic Perspective

Impose structure using symbolic knowledge

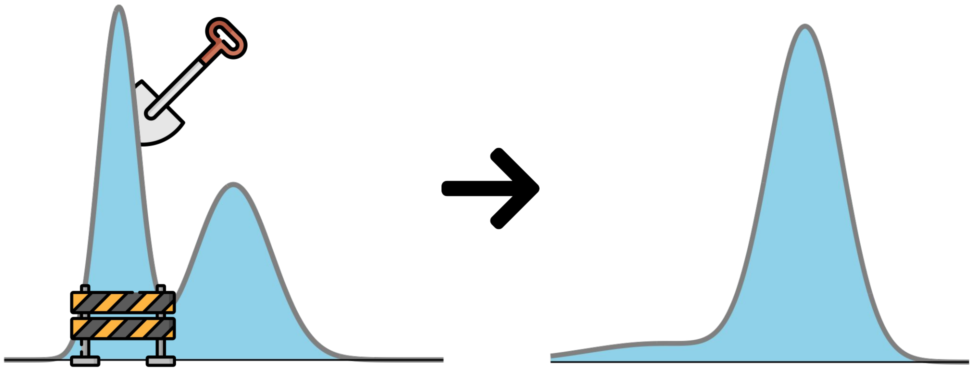


A neural network induces a distribution

[Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang and Guy Van den Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge, *ICML*, 2018]

Neuro-Symbolic AI: A Probabilistic Perspective

Impose structure using symbolic knowledge



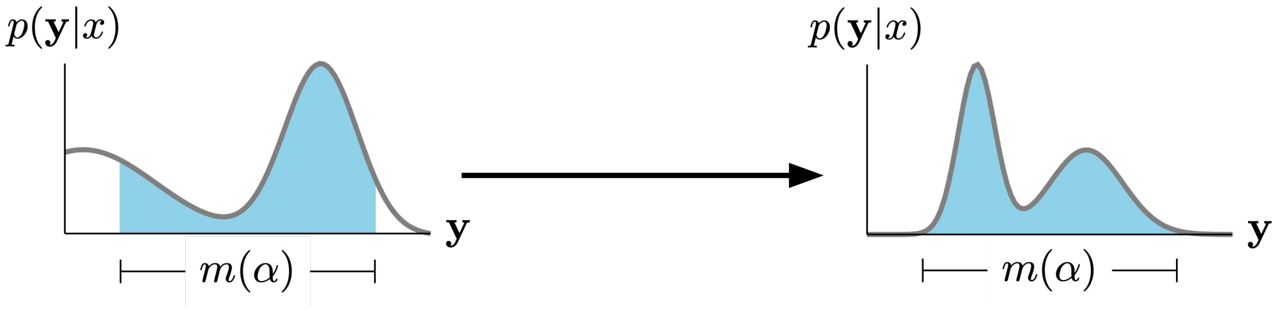
A neural network induces a distribution

Move mass around to be consistent with structure

[Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang and Guy Van den Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge, *ICML*, 2018]

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

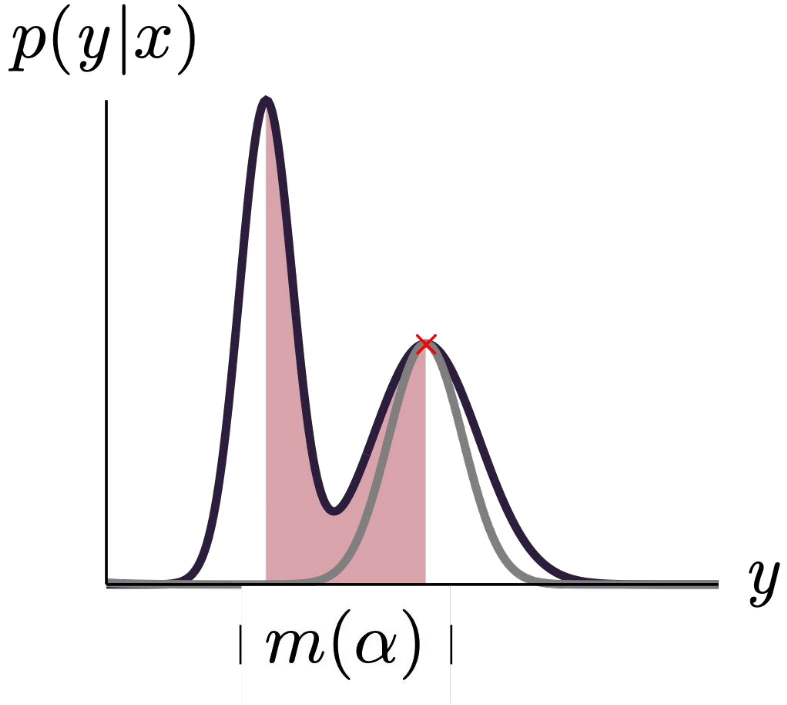
$$p(\boldsymbol{y}) = \prod_{i=1}^{n} p(y_i \mid y_{<i}),$$

[Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang and Guy Van den Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge, *ICML*, 2018]

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.



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*

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

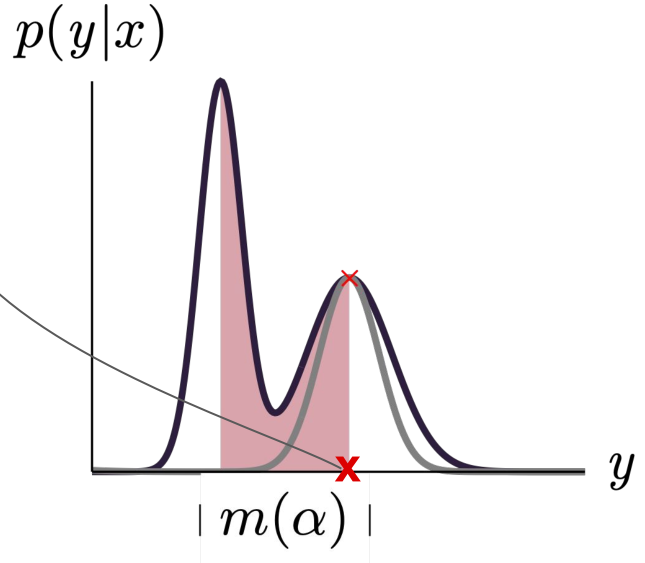
Kareem Ahmed, Kai-Wei Chang and Guy Van den Broeck. A Pseudo-Semantic Loss for Deep Generative Models with Logical Constraints, *In Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

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

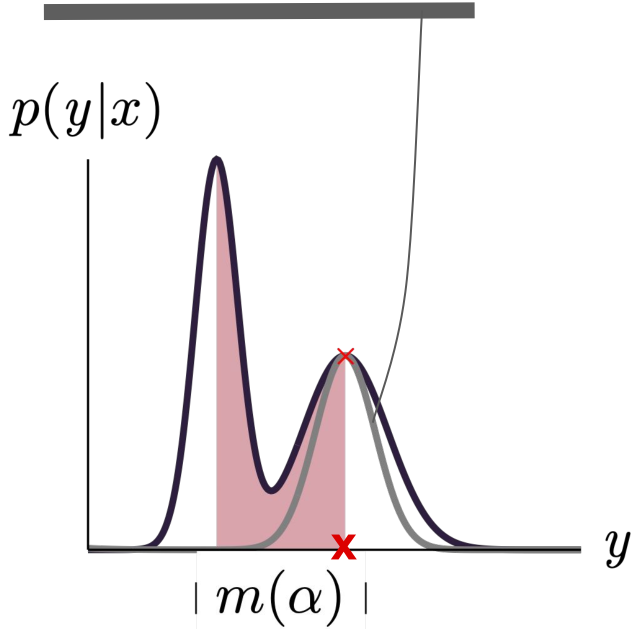


Formally, minimize the *pseudo-semantic loss*

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

Basic Idea:

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



How to compute pseudo-semantic loss?

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

$$\begin{array}{cccccccccc}
 & p(\text{She}) & & p(\text{caught}|\text{I}) & & p(\text{the}|\text{I, saw}) & & p(\text{cat}|\text{I, saw, a}) & & p(\text{yesterday}|\text{I, saw, a, dog}) \\
p(\text{I saw a dog today}) = & p(\text{I}) & \times & p(\text{saw}|\text{I}) & \times & p(\text{a}|\text{I, saw}) & \times & p(\text{dog}|\text{I, saw, a}) & \times & p(\text{today}|\text{I, saw, a, dog}) \\
 & p(\text{He}) & & p(\text{bought}|\text{I}) & & p(\text{an}|\text{I, saw}) & & p(\text{mouse}|\text{I, saw, a}) & & p(\text{tomorrow}|\text{I, saw, a, dog}) \\
 & \cdots & & \cdots & & \cdots & & \cdots & & \cdots
\end{array}$$

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

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.

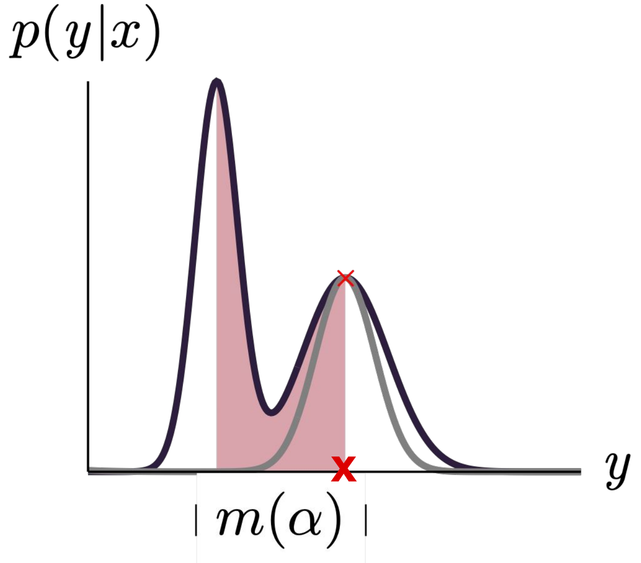
Formally, minimize the *pseudo-semantic loss*

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

x

Basic Idea:

Compute Pr(α) locally and maximize it



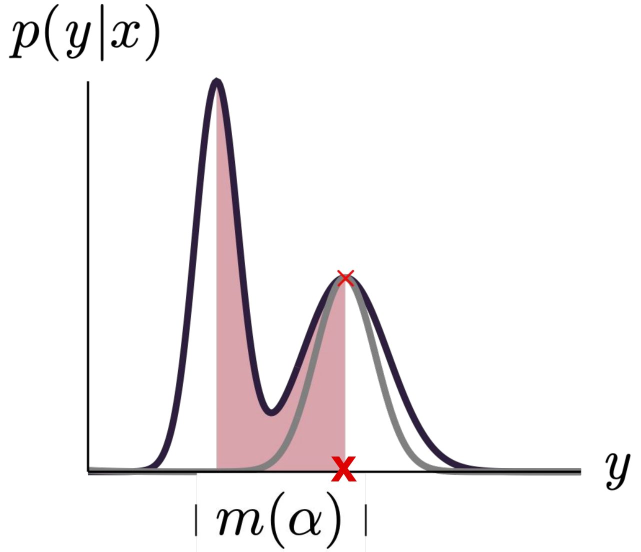
Formally, minimize the *pseudo-semantic loss*

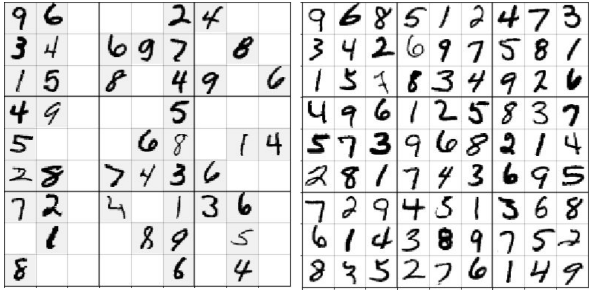
$$\mathcal{L}^{\text{SL}}_{\text{pseudo}} := -\log \mathbb{E}_{\tilde{\boldsymbol{y}} \sim p} \sum_{\boldsymbol{y} \models \alpha} \prod_{i=1}^{n} p(\boldsymbol{y}_i \mid \tilde{\boldsymbol{y}}_{-i})$$

x

How good is this approximation?

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





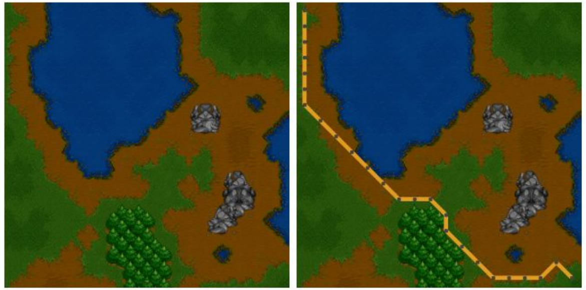


Table 1: Our experimental results on Sudoku.

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. A neurosymbolic problem hidden in LLMs
2. The paradox of learning to reason from data

~~end-to-end learning~~

3. Symbolic reasoning at generation time

4. Symbolic reasoning at training time

logical + probabilistic reasoning + deep learning

Thanks

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

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