UCLA Computer Science

STAR AI
RESEARCH LAB
UCLA

Neuro-Symbolic AI with Tractable Deep Generative Models

Guy Van den Broeck

NeurIPS MATH-AI Workshop - Dec 15 2023

Outline

1. Language generation *with constraints*
2. Structured output learning *with constraints*
3. Autoregressive model learning *with constraints*

Outline

1. **Language generation *with constraints***
2. Structured output learning *with constraints*
3. Autoregressive model learning *with constraints*

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

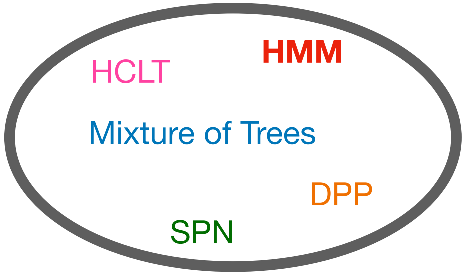
cold	0.50
warm	0.01

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

Marginalization!

Tractable Probabilistic Models

Probabilistic (Generating) Circuits



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

e.g., efficient marginalization:

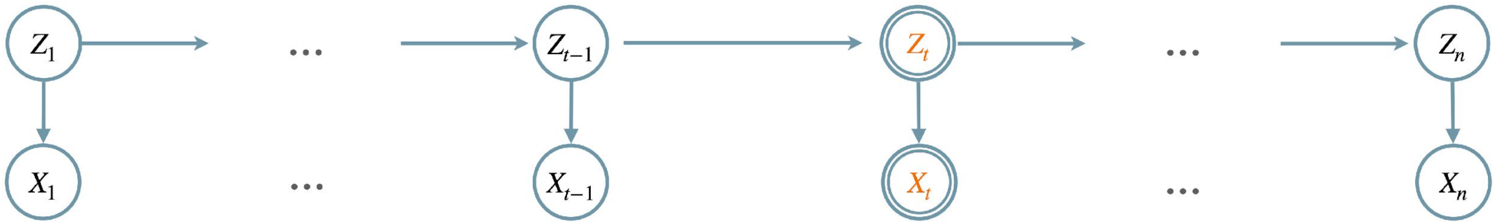
$$p_{TPM}(\text{3rd token} = \text{frisbee}, \text{5th token} = \text{dog})$$

Easily understood as tractable probabilistic circuits.

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 <u>latent variable distillation</u> 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: $(w_{1,1} \lor \ldots \lor w_{1,d1}) \land \ldots \land (w_{m,1} \lor \ldots \lor w_{m,dm})$

Each clause represents the inflections for one keyword.

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

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

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

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}

<u>*Supervised*</u>

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 <u>*weighted geometric mean*</u>:

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

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 guaranteed to be satisfied:
   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} does not depend on α,
   which is only imposed at inference (generation) time.

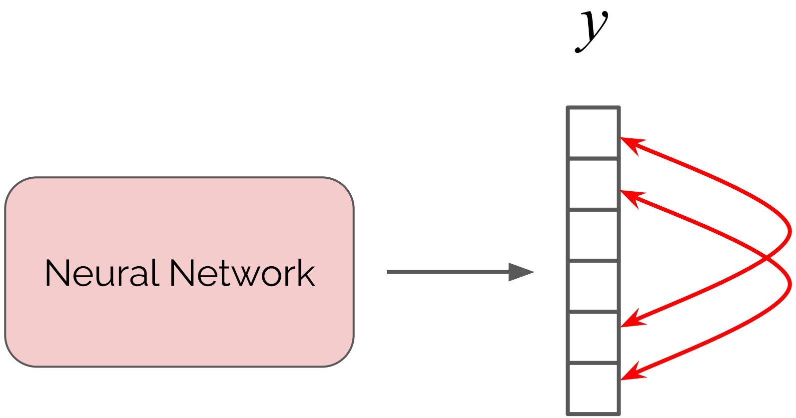
3. Can impose additional tractable constraints:
   - keywords follow a particular order
   - keywords appear at a particular position
   - keywords must not appear

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

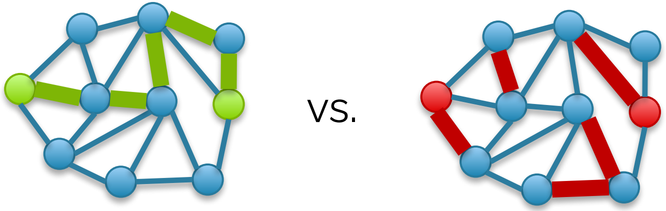
Outline

1. Language generation *with constraints*
2. **Structured output learning *with constraints***
3. Autoregressive model learning *with constraints*

Declarative Knowledge of the Output



How is the output structured?
Are all possible outputs valid?



How are the outputs related to each other?

Learning this from data is inefficient
Much easier to express this declaratively

Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck and Sameer Singh. PYLON: A PyTorch Framework for Learning with Constraints

pylon

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)
```

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck and Sameer Singh. PYLON: A PyTorch Framework for Learning with Constraints

pylon

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)

    loss += constraint_loss(check)(py)
```

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

2 Add as loss to training

```
loss += constraint_loss(check)
```

Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck and Sameer Singh. PYLON: A PyTorch Framework for Learning with Constraints

pylon

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)

    loss += constraint_loss(check)(py)
```

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

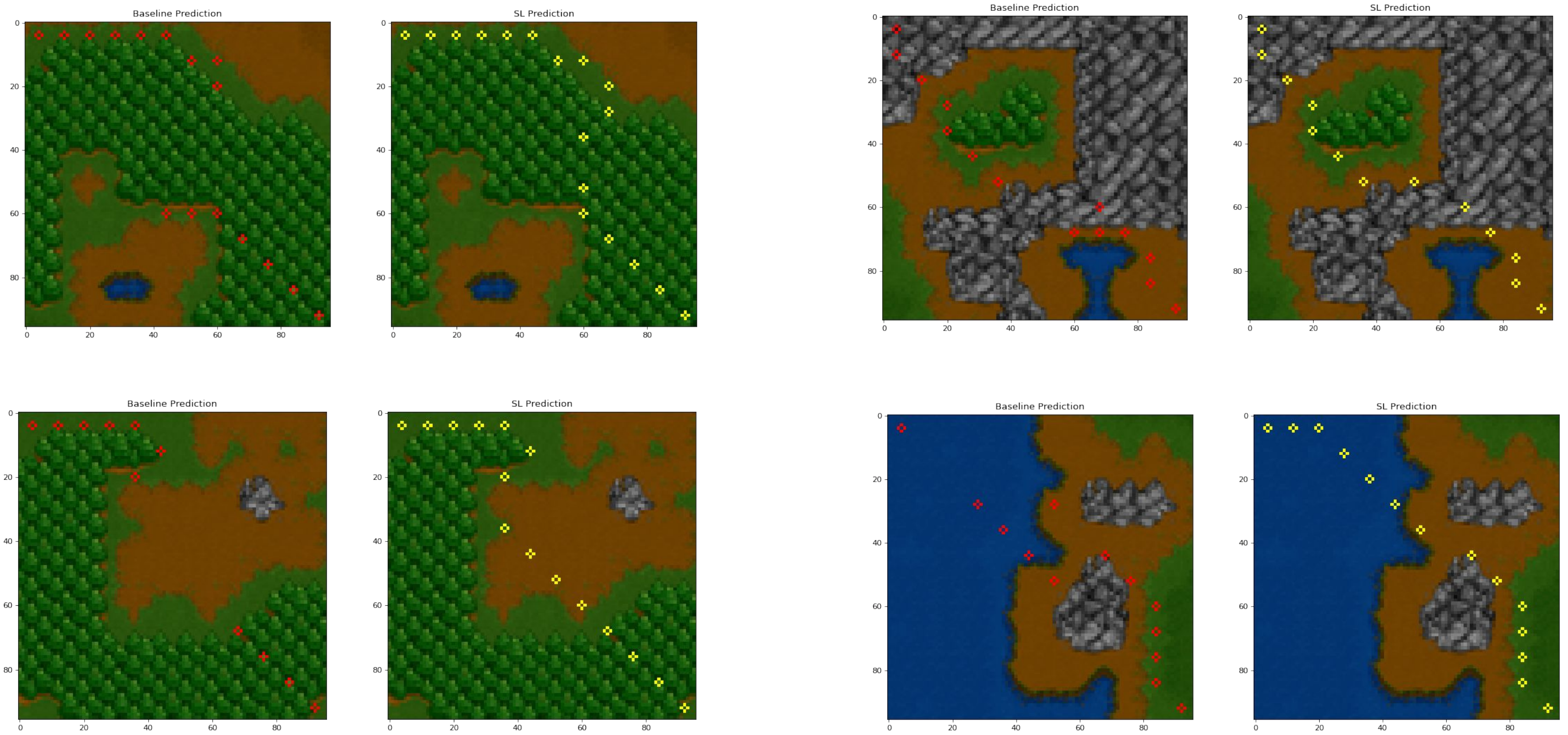
2 Add as loss to training

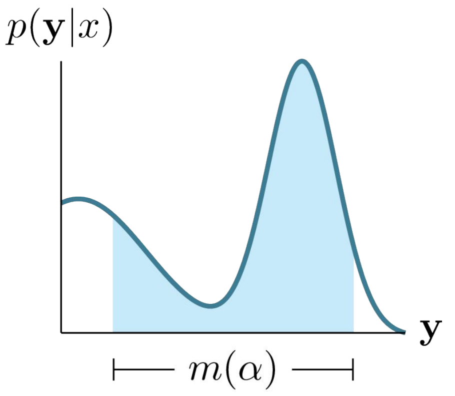
```
loss += constraint_loss(check)
```

3 pylon derives the gradients
(solves a combinatorial problem)

Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck and Sameer Singh. PYLON: A PyTorch Framework for Learning with Constraints

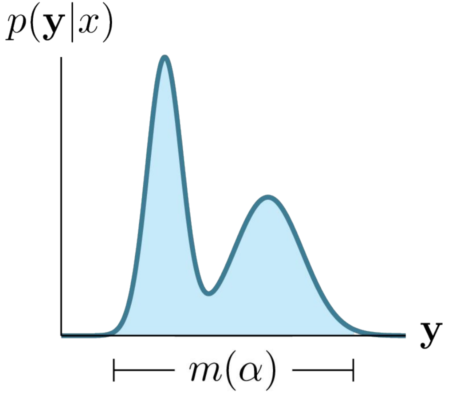
without constraint *with constraint* *without constraint* *with constraint*





a) A network uncertain over both valid & invalid predictions

Semantic Loss



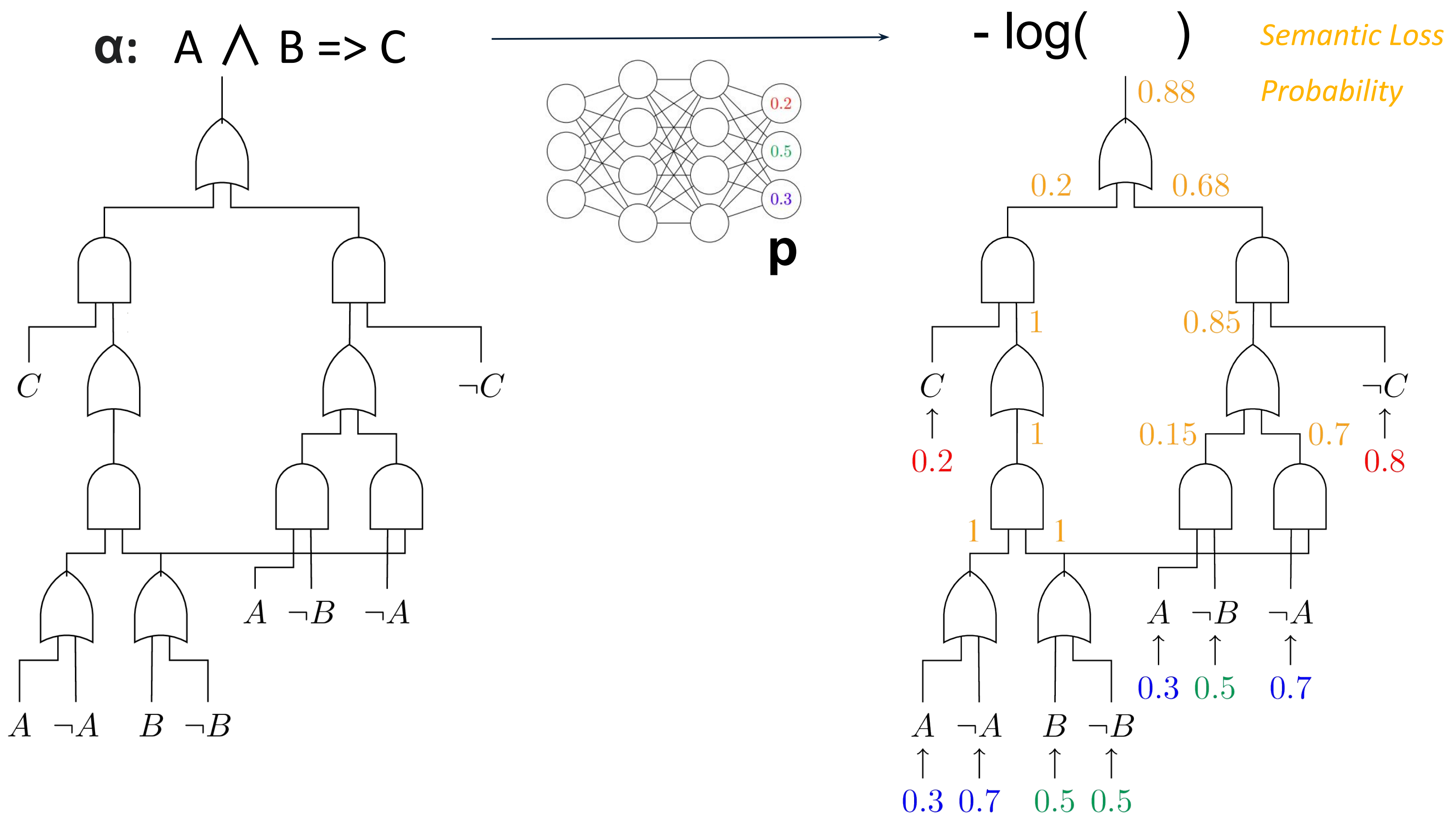
c) A network allocating most of its mass to models of constraint

$$\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \propto -\log \underbrace{\sum_{\mathbf{x} \models \alpha} \prod_{i:\mathbf{x} \models X_i} \mathsf{p}_i \prod_{i:\mathbf{x} \models \neg X_i} (1 - \mathsf{p}_i)}$$

Probability of satisfying constraint α after sampling from neural net output layer **p**

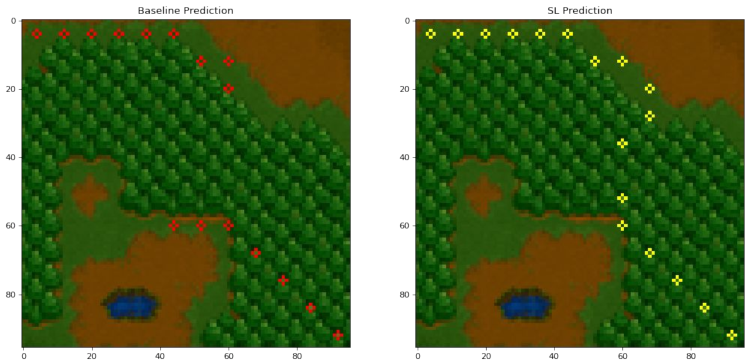
In general: #P-hard ☹️

Do this probabilistic-logical reasoning during learning in a computation graph

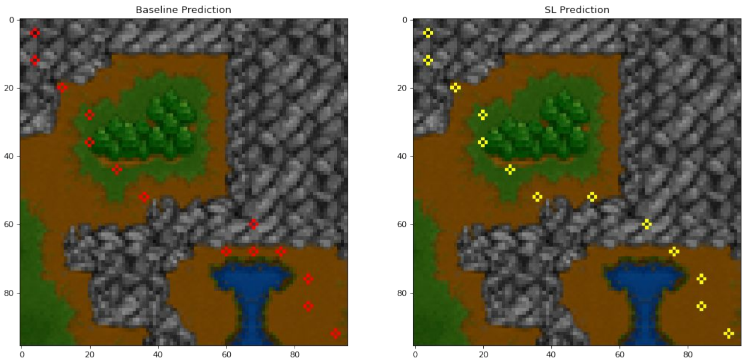


α: A ∧ B => C
- log( )
Semantic Loss
Probability
p
0.2
0.5
0.3
0.88
0.2
0.68
1
0.85
C
¬C
1
0.15
0.7
0.2
0.8
1
1
A
¬B
¬A
0.3
0.5
0.7
A
¬A
B
¬B
0.3
0.7
0.5
0.5

without constraint *with constraint*



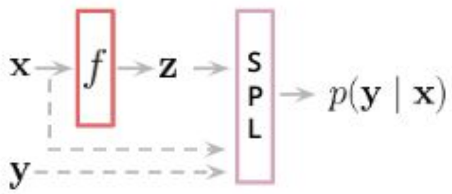
without constraint *with constraint*

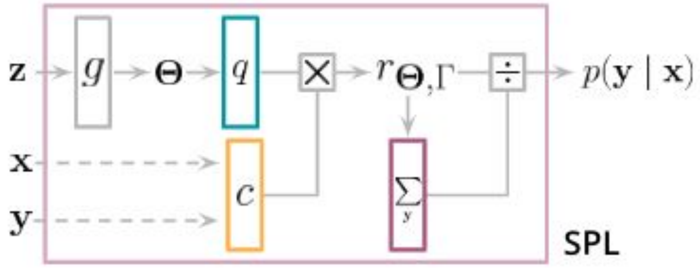


ARCHITECTURE	EXACT MATCH	HAMMING SCORE	CONSISTENCY
RESNET-18+FIL	55.0	**97.7**	56.9
RESNET-18+$\mathcal{L}_{\mathrm{SL}}$	59.4	**97.7**	61.2

Semantic Probabilistic Layers

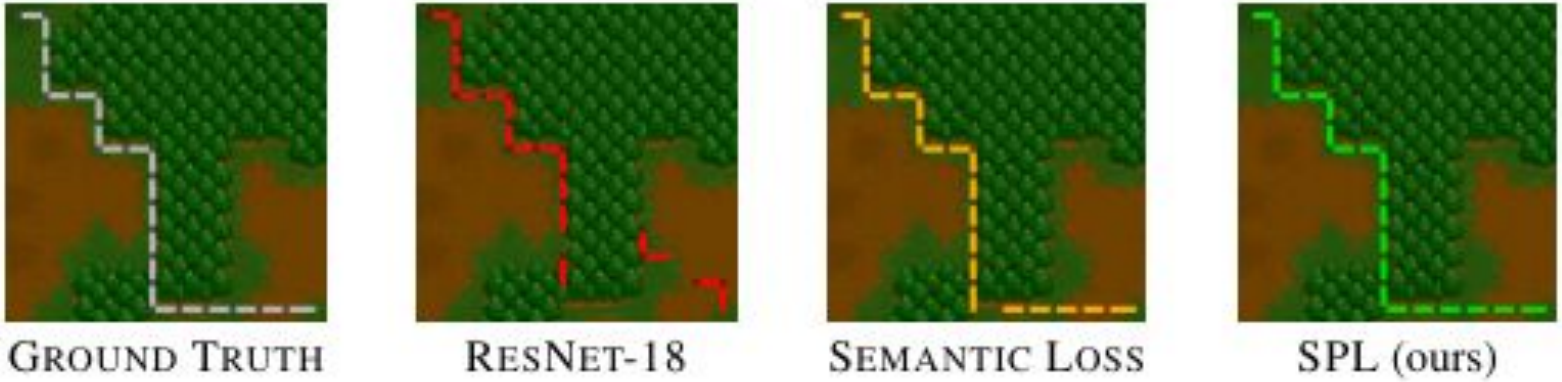
- How to give a 100% guarantee that Boolean constraints will be satisfied?
- Bake the constraint into the neural network as a special layer





- Secret sauce is again tractable circuits – computation graphs for reasoning

Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck and Antonio Vergari. Semantic Probabilistic Layers for Neuro-Symbolic Learning, 2022.



ARCHITECTURE	EXACT MATCH	HAMMING SCORE	CONSISTENCY
RESNET-18+FIL	55.0	**97.7**	56.9
RESNET-18+$\mathcal{L}_{\mathsf{SL}}$	59.4	**97.7**	61.2
RESNET-18+SPL	75.1	97.6	**100.0**
OVERPARAM. SPL	**78.2**	96.3	**100.0**

Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck and Antonio Vergari. Semantic Probabilistic Layers for Neuro-Symbolic Learning, 2022.

Outline

1. Language generation *with constraints*
2. Structured output learning *with constraints*
3. **Autoregressive model learning *with constraints***

Autoregressive distributions are hard…

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

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

Autoregressive distributions are hard…

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

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

Why did it work before?

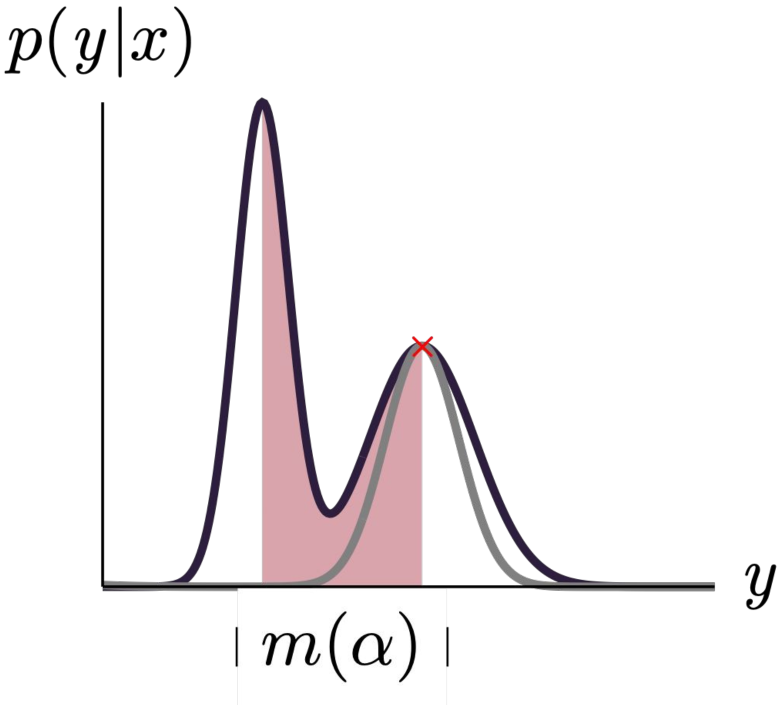
$$\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \propto -\log \underbrace{\sum_{\mathbf{x} \models \alpha} \prod_{i:\mathbf{x} \models X_i} \mathsf{p}_i \prod_{i:\mathbf{x} \models \neg X_i} (1 - \mathsf{p}_i)}$$

Probability of satisfying constraint α
after sampling from neural net output layer **p**
ASSUMING INDEPENDENT BERNOULLI'S

Basic Idea:

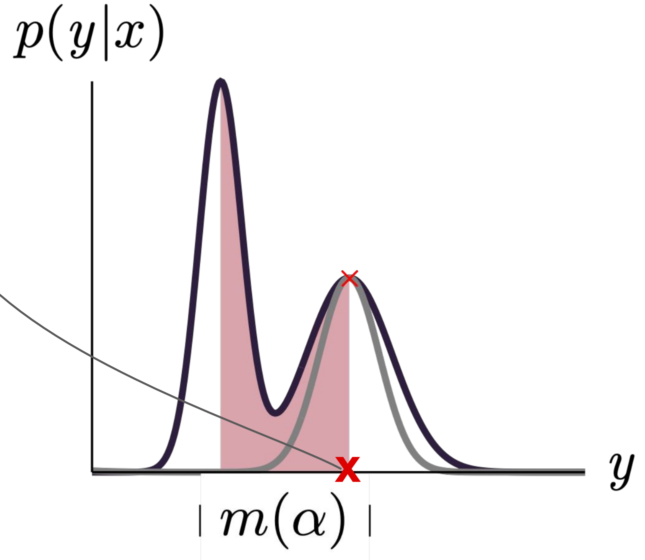
Use how likely a constraint is to be satisfied around a model sample (x) as a proxy for how likely it is to be satisfied under the entire distribution.

Average over many such samples.



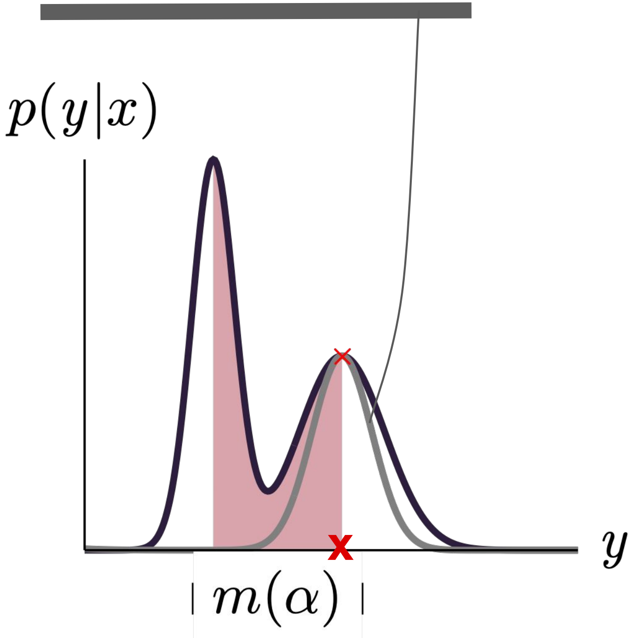
Formally, minimize the *pseudo-semantic loss*

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



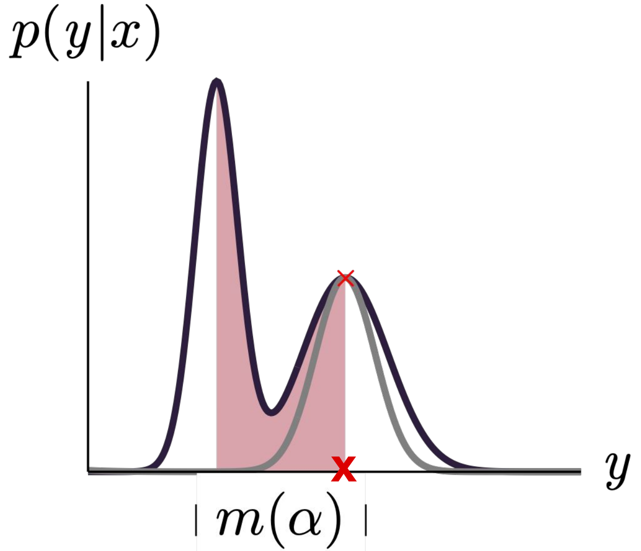
Formally, minimize the *pseudo-semantic loss*

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



Formally, minimize the *pseudo-semantic loss*

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



How good is this approximation?

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

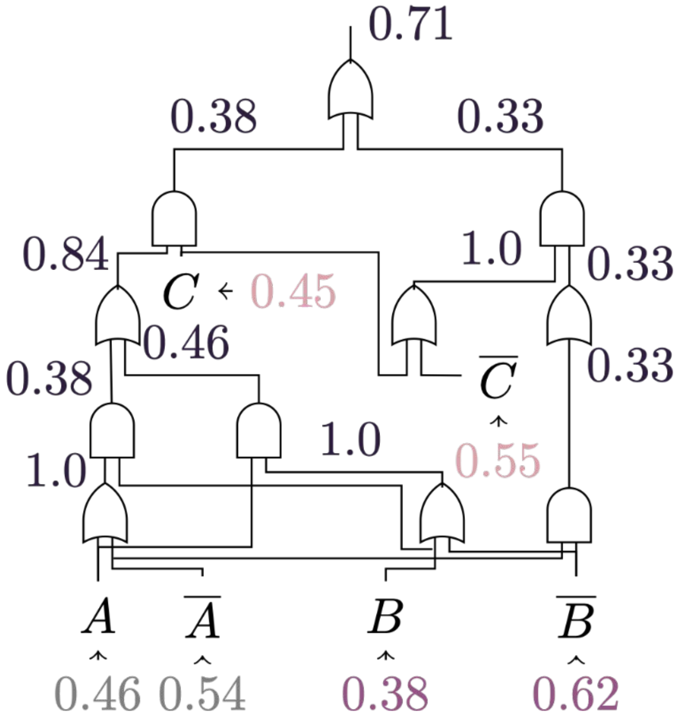
How to compute pseudo-semantic loss?

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

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

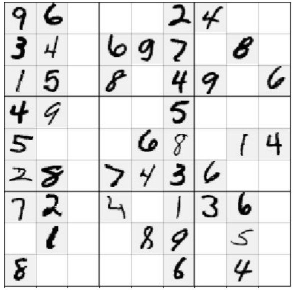
$$\rightarrow \begin{cases} p(abc) = 0.13 & p(abc) = 0.13 & p(abc) = 0.13 \\ p(\bar{a}bc) = 0.15 & p(a\bar{b}c) = 0.21 & p(ab\bar{c}) = 0.16 \end{cases}$$

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



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**



Detoxify LLMs by disallowing bad words

Constraint α is a list of 403 toxic words
Evaluation is a toxicity classifier

Models	Avg. Toxicity (↓) Full	Toxic	Nontoxic
GPT-2	0.11 ± 0.15	0.69 ± 0.13	0.09 ± 0.19
GPT-2 + NeuroLogic [25]	0.08 ± 0.14	0.66 ± 0.13	$\mathbf{0.06 \pm 0.08}$
GPT-2 + Word Banning	0.12 ± 0.16	0.69 ± 0.13	0.09 ± 0.11
PseudoSL	$\mathbf{0.06 \pm 0.09}$	0.59 ± 0.04	$\mathbf{0.06 \pm 0.08}$
PseudoSL + NeuroLogic [25]	$\mathbf{0.05 \pm 0.10}$	0.68 ± 0.15	$\mathbf{0.05 \pm 0.07}$
PseudoSL + Word Banning	$\mathbf{0.06 \pm 0.09}$	$\mathbf{0.58 \pm 0.01}$	$\mathbf{0.06 \pm 0.08}$

Detoxify LLMs by disallowing bad words

Constraint α is a list of 403 toxic words
Evaluation is a toxicity classifier

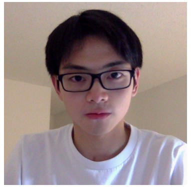
Models		Avg. Toxicity (↓) Full	Toxic	Nontoxic	Valid. PPL
Domain-Adaptive Training	GPT-2	0.12 ± 0.15	0.67 ± 0.12	0.10 ± 0.11	24.52
	SGEAT	$\mathbf{0.07 \pm 0.09}$	0.64 ± 0.11	$\mathbf{0.06 \pm 0.08}$	25.93
	PseudoSL	$\mathbf{0.07 \pm 0.09}$	$\mathbf{0.61 \pm 0.09}$	0.07 ± 0.09	26.60

Outline

1. Language generation *with constraints*
2. Structured output learning *with constraints*
3. Autoregressive model learning *with constraints*

Thanks

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



Honghua

Kareem

Zhe

Meihua

Anji

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