Symbolic Reasoning for Large Language Models

Guy Van den Broeck

TU Dortmund AI Colloquium - Jun 13 2024
Outline

1. The paradox of learning to reason from data
   
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.

Unclear whether they follow the rules of logical deduction.

**Language Models:**

\[
\text{input} \rightarrow ? \rightarrow \text{Carol is the grandmother of Justin}.
\]

**Logical Reasoning:**

\[
\text{input} \rightarrow \text{Justin in Kristin’s son; Carol is Kristin’s mother;} \rightarrow \text{Carol is Justin’s mother’s mother;} \rightarrow \text{if } X \text{ is } Y\text{’s mother’s mother then } X \text{ is } Y\text{’s grandmother} \rightarrow \text{Carol is the grandmother of Justin}.
\]
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

Facts:
- Alice is *fast*.
- Alice is *normal*.

Rules:
- If Alice is *fast* and *smart*, then Alice is *bad*.
- If Alice is *normal*, then Alice is *smart*.
- If Alice is *normal* and *happy*, then Alice is *sad*.

Query 1: Alice is *bad*. [Answer: True]
Query 2: Alice is *sad*. [Answer: False]
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.

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

Rule-Priority

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

<table>
<thead>
<tr>
<th>Test</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP</td>
<td>99.9</td>
<td>99.8</td>
<td>99.7</td>
<td>99.3</td>
<td>98.3</td>
<td>97.5</td>
<td>95.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>100.0</td>
<td>100.0</td>
<td>99.9</td>
<td>99.9</td>
<td>99.7</td>
<td>99.7</td>
<td>99.0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Train</th>
<th>Test</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP</td>
<td>RP</td>
<td>99.9</td>
<td>99.8</td>
<td>99.7</td>
<td>99.3</td>
<td>98.3</td>
<td>97.5</td>
<td>95.5</td>
</tr>
<tr>
<td>RP</td>
<td>LP</td>
<td>99.8</td>
<td>99.8</td>
<td>99.3</td>
<td>96.0</td>
<td>90.4</td>
<td>75.0</td>
<td>57.3</td>
</tr>
<tr>
<td>LP</td>
<td>RP</td>
<td>97.3</td>
<td>66.9</td>
<td>53.0</td>
<td>54.2</td>
<td>59.5</td>
<td>65.6</td>
<td>69.2</td>
</tr>
<tr>
<td>LP</td>
<td>LP</td>
<td>100.0</td>
<td>100.0</td>
<td>99.9</td>
<td>99.9</td>
<td>99.7</td>
<td>99.7</td>
<td>99.0</td>
</tr>
</tbody>
</table>

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 \( \Pr(\text{label} = \text{True} | \text{rule#} = x) = 0.5 \) for all \( x \)

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Test</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP</td>
<td>99.9</td>
<td>99.8</td>
<td>99.7</td>
<td>99.3</td>
<td>98.3</td>
<td>97.5</td>
<td>95.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP_b</td>
<td>99.0</td>
<td>99.3</td>
<td>98.5</td>
<td>97.5</td>
<td>96.7</td>
<td>93.5</td>
<td>88.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

Here's the correct sentence: The **dog caught** the **frisbee** in mid-air, showing off its amazing catching skills.
A frisbee is caught by a dog.
A pair of frisbee players are caught in a dog fight.
What do we have?

Prefix: “The weather is”

Constraint $\alpha$: text contains “winter”

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

<table>
<thead>
<tr>
<th>cold</th>
<th>0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>warm</td>
<td>0.10</td>
</tr>
</tbody>
</table>

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 $\alpha$: text contains “winter”

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

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

Marginalization!
Tractable Probabilistic Models

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

e.g., efficient marginalization:

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

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

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

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

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 $\alpha$ in CNF: $(w_{1,1} \lor \ldots \lor w_{1,d_1}) \land \ldots \land (w_{m,1} \lor \ldots \lor w_{m,d_m})$

Each clause represents the inflections for one keyword.
Computing $p(\alpha \mid x_{1:t+1})$

For constraint $\alpha$ in CNF:

$$(w_{1,1} \lor \ldots \lor w_{1,d_1}) \land \ldots \land (w_{m,1} \lor \ldots \lor w_{m,d_m})$$

e.g., $\alpha = ("swims" \lor "like swimming") \land ("lake" \lor "pool")$

**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
**Overview**

**Lexical Constraint** \( \alpha \): sentence contains keyword "winter"

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

---

[Diagram showing pre-trained language model and tractable probabilistic model, minimizing 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** $\alpha$: sentence contains keyword “winter”

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

- **Intractable**
  - $x_{t+1}$: cold 0.05
  - $x_{t+1}$: warm 0.10

- **Efficient**
  - $x_{t+1}$: cold 0.50
  - $x_{t+1}$: warm 0.01

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

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

\[ p_{hmm}(\text{next-token} | \text{prefix}, \alpha) \propto p_{hmm}(\alpha | \text{next-token}, \text{prefix}) \cdot p_{lm}(\text{next-token} | \text{prefix}) \]

<table>
<thead>
<tr>
<th>Method</th>
<th>ROUGE-L</th>
<th>Generation Quality</th>
<th>Constraint Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev test</td>
<td>BLEU-4  test</td>
<td>SPICE  test</td>
</tr>
<tr>
<td>Unsupervised</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InsNet (Lu et al., 2022a)</td>
<td>- -</td>
<td>18.7 -</td>
<td>- -</td>
</tr>
<tr>
<td>NeuroLogic (Lu et al., 2021)</td>
<td>- 41.9</td>
<td>- 24.7 -</td>
<td>- 14.4 -</td>
</tr>
<tr>
<td>A*esque (Lu et al., 2022b)</td>
<td>- 44.3</td>
<td>- 28.6 -</td>
<td>- 15.6 -</td>
</tr>
<tr>
<td>NADÔ (Meng et al., 2022)</td>
<td>- -</td>
<td>26.2 -</td>
<td>- -</td>
</tr>
<tr>
<td>GeLaTo</td>
<td>44.6 44.1</td>
<td>29.9 29.4</td>
<td>16.0 15.8</td>
</tr>
</tbody>
</table>

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*:

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

<table>
<thead>
<tr>
<th>Method</th>
<th>ROUGE-L</th>
<th>Generation Quality</th>
<th>Constraint Satisfaction</th>
<th>Constraint Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev</td>
<td>test</td>
<td></td>
<td>Coverage</td>
</tr>
<tr>
<td><strong>Supervised</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NeuroLogic (Lu et al., 2021)</td>
<td>-</td>
<td>42.8</td>
<td>-</td>
<td>14.7</td>
</tr>
<tr>
<td>A*esque (Lu et al., 2022b)</td>
<td>-</td>
<td>43.6</td>
<td>-</td>
<td>15.2</td>
</tr>
<tr>
<td>NADO (Meng et al., 2022)</td>
<td>44.4†</td>
<td>-</td>
<td>30.8</td>
<td>16.1†</td>
</tr>
<tr>
<td>GeLaTo</td>
<td>46.0</td>
<td>45.6</td>
<td>34.1</td>
<td>32.9†</td>
</tr>
</tbody>
</table>

Advantages of GeLaTo:

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

2. Training $p_{hmm}$ does not depend on $\alpha$, 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.
Tractable Control with Ctrl-G

conditioning on logical constraints...

1. distill

2. classical algorithms

HMM

DFA

3. inference-time guidance

outputs satisfying the 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.”

```python
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.”
Tractable Control with Ctrl-G

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

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 $\alpha$ (a list of 403 toxic words not to say)
2. training data $D$

Learn: a transformer $Pr(.)$ that

1. satisfies the constraint $\alpha$: $Pr(\alpha)\uparrow$
2. maximizes the likelihood: $Pr(D)\uparrow$
Neurosymbolic learning of transformers

Given:

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

Learn: a transformer $Pr(.)$ that

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

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

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

\[
\mathcal{L}_{\text{pseudo}}^{SL} := - \log \mathbb{E}_{\tilde{y} \sim p} \sum_{y \models \alpha} \prod_{i=1}^{n} p(y_i \mid \tilde{y}_{-i})
\]
Formally, minimize the pseudo-semantic loss

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

Basic Idea:
Pick a location to build the approximation around
Formally, minimize the *pseudo-semantic loss*

\[ \mathcal{L}_{\text{pseudo}}^{SL} := - \log \mathbb{E}_{\tilde{y} \sim p} \sum_{x \in \mathcal{X}} \prod_{i=1}^{n} p(y_i \mid \tilde{y}_{-i}) \]

**Basic Idea:**
Extract a local tractable probabilistic model around the point (independent in each dimension)
Formally, minimize the *pseudo-semantic loss*

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

**Basic Idea:**

Compute \( \Pr(\alpha) \) locally and maximize it.
Formally, minimize the pseudo-semantic loss

\[ \mathcal{L}_{\text{pseudo}}^{SL} := - \log \mathbb{E}_{\tilde{y} \sim p} \sum_{y|x=\alpha} \prod_{i=1}^{n} p(y_i | \tilde{y}_{-i}) \]

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?

\[ p_\theta \sim abc \]
How to compute pseudo-semantic loss?

\[ p_\theta \sim abc \]

\[ \rightarrow \{ \begin{array}{c}
abc & abc & abc \\
\bar{abc} & \bar{abc} & \bar{abc} \\
abla \\
\end{array} \]
How to compute pseudo-semantic loss?

\[ p_\theta \sim abc \]

\[
\begin{align*}
&\Rightarrow \left\{ \ight. \\
&\quad abc \quad abc \quad abc \\
&\quad \bar{abc} \quad \bar{abc} \quad a\bar{c} \\
&\Rightarrow \left\{ \ight. \\
&\quad p(abc) = 0.13 \quad p(abc) = 0.13 \quad p(abc) = 0.13 \\
&\quad p(\bar{abc}) = 0.15 \quad p(a\bar{bc}) = 0.21 \quad p(ab\bar{c}) = 0.16 \\
&\Rightarrow \left\{ \ight. \\
&\quad p(a|bc) = 0.46 \quad p(b|ac) = 0.38 \quad p(c|ab) = 0.45 \\
&\quad p(\bar{a}|bc) = 0.54 \quad p(\bar{b}|ac) = 0.62 \quad p(\bar{c}|ab) = 0.55
\end{align*}
\]
How to compute pseudo-semantic loss?

\[ p_\theta \sim abc \]

\[
\begin{align*}
  &\{ abc \quad abc \quad abc \\
  \rightarrow &\quad \bar{abc} \quad \bar{abc} \quad \bar{abc} \\
  &\{ p(abc) = 0.13 \quad p(abc) = 0.13 \quad p(abc) = 0.13 \\
  \rightarrow &\quad p(\bar{abc}) = 0.15 \quad p(\bar{abc}) = 0.21 \quad p(\bar{abc}) = 0.16 \\
  &\{ p(a|bc) = 0.46 \quad p(b|ac) = 0.38 \quad p(c|ab) = 0.45 \\
  \rightarrow &\quad p(\bar{a}|bc) = 0.54 \quad p(\bar{b}|ac) = 0.62 \quad p(\bar{c}|ab) = 0.55
\end{align*}
\]
### Table 1: Our experimental results on Sudoku.

<table>
<thead>
<tr>
<th>Test accuracy %</th>
<th>Exact</th>
<th>Consistent</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvNet</td>
<td>16.80</td>
<td>16.80</td>
</tr>
<tr>
<td>ConvNet + SL</td>
<td>22.10</td>
<td>22.10</td>
</tr>
<tr>
<td>RNN</td>
<td>22.40</td>
<td>22.40</td>
</tr>
<tr>
<td>RNN + PSEUDO SL</td>
<td><strong>28.20</strong></td>
<td><strong>28.20</strong></td>
</tr>
</tbody>
</table>

### Table 2: Our experimental results on Warcraft.

<table>
<thead>
<tr>
<th>Test accuracy %</th>
<th>Exact</th>
<th>Consistent</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>55.00</td>
<td>56.90</td>
</tr>
<tr>
<td>ResNet-18 + SL</td>
<td>59.40</td>
<td>61.20</td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>62.00</td>
<td>76.60</td>
</tr>
<tr>
<td>CNN-LSTM + PSEUDO SL</td>
<td><strong>66.00</strong></td>
<td><strong>79.00</strong></td>
</tr>
</tbody>
</table>

Detoxify LLMs by disallowing bad words

Constraint $\alpha$ is a list of 403 toxic words not to say.
Evaluation is a toxicity classifier.

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

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!

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