Prompting is Not All You Need


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.

ChatGPT fails to follow simple logical constraints!

Constrained Generation is Challenging

Logical Constraint 2: e.g., text contains keyword “winter”

Constrained Autoregressive Generation. Our goal is to generate from

\[ p(x_{1:n} | w) = \prod_{i=1}^{n} p(x_i | x_{1:i-1}, w) \]

at each step, e.g., suppose we have generated the first n tokens x_{1:n} as “the weather is”; then we generate the next token x_{n+1} from transfer

\[ p(x_{n+1} | x_{1:n}, w) \]

Advantages of GeLaTo

1. Logical constraint is guaranteed to be satisfied. When generating next token x_{n+1}, if x_{n+1} is not included, then \[ p(x_{n+1} | x_{1:n}, w) = 0 \]; hence \( w \) will not be generated.

2. The training of \( P_{\text{inum}} \) does not depend on \( w \), which is only imposed during generation. Once \( P_{\text{inum}} \) is trained, GeLaTo generalizes to any tractable constraints.

GeLaTo

We propose GeLaTo (Generating Language with Tractable Constraints) to guide autoregressive generation from LLMs. Tractable Probabilistic Models (TPMs) are generative models \( P_{\text{inum}}(x_{1:n}) \) that allow efficient conditioning. We use hidden Markov models (HMMs) as an example.

Step 1. Distilling an HMM from LM

Train \( P_{\text{inum}} \) on \( D \) to minimize their KL-divergence.

Step 2. Probabilistic Reasoning with Constraints

Compute \( P_{\text{inum}}(x_{1:n+1} | x_{1:n}, w) \) to approximate \( p(x_{1:n+1} | x_{1:n}, w) \), then generate from:

\[ P_{\text{inum}}(x_{1:n+1} | x_{1:n}, w) \propto P_{\text{inum}}(x_{1:n} | x_{1:n}, w) \cdot P_{\text{inum}}(w) \]

GeLaTo can also help prompting!

Given some prompt \( w \) that represents \( a \), e.g., “keywords = ‘XXX’”, we can combine \( P_{\text{inum}} \) and \( p_t \) by taking their weighted geometric mean:

\[ P_{\text{inum}}(x_{1:n} | x_{1:n}, a) \propto P_{\text{inum}}(x_{1:n} | x_{1:n}, w) \cdot P_t(x_i = w | x_i) \]

GeLaTo can enforce various logical constraints

1. Keywords appear (in any order/form of inclusions)
2. (Some) keywords are generated following a specific order.
3. (Some) keywords must appear at specified positions.
4. (Some) keywords must not appear in the generated text.

Probabilistic Reasoning with Constraints

Consider a logical constraint \( w \) encoded as:

\[ (w_1 \land \ldots \land w_n) \land (\lnot (w_{n+1} \lor \ldots \lor w_m)) \]

each \( w_j \) is a string of tokens (“keywords”) that must appear \( \alpha = \{\text{swims} \sim \text{like swimming}\} \)  \( \sim \{\text{lake} \sim \text{pool}\} \)

Need to compute \( P_{\text{inum}}(x_{1:n} | x_{1:n}, w) \) to enforce \( \alpha \).

Efficient PMFs for HMMs

HMMs define distributions over \( x_{1:n} \) and latent variables \( z_{1:n} \):

\[ p(x_{1:n}) = \prod_{t=1}^{n} p(z_t | z_{t-1}, x_{1:t-1}) \cdot p(x_t | z_t, x_{1:t-1}) \]

Assume \( \alpha \) contains single-token keywords, then we can compute \( p(x_{1:n}) \), as well as \( p(x_{1:t}, a) \), by:

\[ p(x_{1:t} | z_{1:t}) = \sum_{z_{t+1:n}} p(x_t | z_t, x_{1:t-1}) \cdot p(z_{t+1:n} | z_{1:t}) \]

The time complexity for sampling from \( P_{\text{inum}}(x_{1:n} | 0) = O(2^n) \).

Experiments

Commonsense Generation (CommonGen)

Input Concepts: snow, car, drive
Output 1: The car drives down a snow-covered road.
Output 2: Driving through the snow, the car crashed.

Table 4. BLEU-4 scores for Yelp!Review and News datasets

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
<th>Yelp!Review</th>
<th>News</th>
<th>NADO</th>
<th>GeLaTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>InsNet</td>
<td>-</td>
<td>58</td>
<td>5.0</td>
<td>4.5</td>
<td>6.6</td>
</tr>
<tr>
<td>NADO</td>
<td>-</td>
<td>58</td>
<td>5.0</td>
<td>4.5</td>
<td>6.6</td>
</tr>
<tr>
<td>GeLaTo</td>
<td>-</td>
<td>58</td>
<td>5.0</td>
<td>4.5</td>
<td>6.6</td>
</tr>
</tbody>
</table>

Table 1. Automatic evaluation results on CommonGen.

References: