Constrained text generation through discrete & continuous inference
Neural text generation

- **Large-scale language models** drive state-of-the-art performance in text generation tasks:

  - **Open-Ended Generation**
    - Build next-gen apps with OpenAI’s powerful models.
    - OpenAI’s API provides access to GPT-3, which performs a wide variety of natural language tasks, and Codex, which translates natural language to code.

  - **Long-form QA**
    - How has technological growth increased so exponentially in the last 50 years?
    - There are many explanations for the exponential growth in technology in the last century. One explanation is that the pace of technological innovation speeds up exponentially over time because of a common force that pushes it forward. Another explanation is that each new generation of technology stands on the shoulders of its predecessors, allowing for improvements that lead to the next generation of even better technology.

  - **Machine Translation**
    - Google Translate
      - Text translation

  - **Program Synthesis**
    - Your AI pair programmer
      - GitHub Copilot (Chen et al. 2021)

  - **Dialogue**
    - Hello, I am a friendly dialogue model. What do you want to talk about?
      - Well, I’ve been wondering about something that you just said.

[Thoppilan et al. 2022]
Neural text generation

• General purpose:

  What is a language model?

  <start> Generate a question. Question:

  Arbitrary “prompt”
Neural text generation

• General purpose:

  What is a language model?

  Internet Train

  Language Model

  <start> Generate a question. Question:

  Arbitrary “prompt”

• Task-specific:

  Jupiter is the fifth planet from the sun.

  fine-tune

  Summarization Language Model

  <start> <end>
• **GPT-3:** a *general purpose* 175B parameter language model:

```
Input
Generate a question.
Output
Question: What is the difference between a covalent bond and an ionic bond?
```

Example from: https://beta.openai.com/playground
Jupiter is the fifth planet from the Sun. It is very large compared to other planets and is one of the brightest objects in the night sky. People have been observing Jupiter since prehistoric times.
Controlling neural text generation

• Controlling the syntax, semantics, or style of generated text is difficult

• Lexical content

Example from: https://beta.openai.com/playground
Controlling neural text generation

- Controlling the syntax, semantics, or style of generated text is difficult

Example based on: https://beta.openai.com/playground/p/default-summarize
Controlling neural text generation

• Controlling the syntax, semantics, or style of generated text is difficult

Summarize this for a second-grade student, and include the word Venus:

Jupiter

From Wikipedia, the free encyclopedia

This article is about the planet. For the Roman god, see Jupiter (mythology). For other uses, see Jupiter (disambiguation).

Jupiter is the fifth planet from the Sun and the largest in the Solar System. It is a gas giant with a mass more than two and a half times that of all the other planets in the Solar System combined, but slightly less than one-thousandth the mass of the Sun. Jupiter is the third brightest natural object in the Earth's night sky after the Moon and Venus. People have been observing it since prehistoric times; it was named after the Roman god Jupiter, the king of the gods, because of its observed

• For a task specific model: how do we even specify the control words?

Example based on: https://beta.openai.com/playground/p/default-summarize
Controlling neural text generation

• Typical usage pattern: use an “off-the-shelf” model to generate text
Controlling neural text generation

• Typical usage pattern: use an “off-the-shelf” model to generate text
• Hard to get data for desired control outcomes
Controlling neural text generation

• Typical usage pattern: use an “off-the-shelf” model to generate text
  • Hard to get data for desired control outcomes
  • Expensive to fine-tune & store a new model
Controlling neural text generation

• Typical usage pattern: use an “off-the-shelf” model to generate text
  • Hard to get data for desired control outcomes
  • Expensive to fine-tune & store a new model
  • How do we enable controlled generation for off-the-shelf models?
Controlling neural text generation

• Typical usage pattern: use an “off-the-shelf” model to generate text
  • Hard to get data for desired control outcomes
  • Expensive to fine-tune & store a new model

• How do we enable controlled generation for off-the-shelf models?
  • General-purpose or task-specific
Control through inference
Model + decoding
Control through inference

Model + decoding

- Text generation involves two steps:
Control through inference
Model + decoding

• Text generation involves two steps:
• Learn a model from data (or download one…)

\[ p_\theta(y \mid x) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, x) \]
Control through inference
Model + decoding

- Text generation involves two steps:
- Learn a model from data (or download one…)

\[ p_\theta(y \mid x) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, x) \]

- Use an inference/decoding algorithm to generate text
- \( \hat{y} = \text{decode}(p_\theta(\cdot \mid x)) \)
Control through inference
Model + decoding

- Text generation involves two steps:
- Learn a **model** from data (or download one...)

\[ p_\theta(y \mid x) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, x) \]

- Use an **inference/decoding** algorithm to generate text
- \( \hat{y} = \text{decode}(p_\theta( \cdot \mid x)) \)
  - e.g. sampling, \( y_t \sim p_\theta(y_t \mid y_{<t}, x) \)

What is the mass of Jupiter?
Control through inference
Model + decoding

- Text generation involves two steps:
- Learn a model from data (or download one…)

\[ p_\theta(y | x) = \prod_{t=1}^{T} p_\theta(y_t | y_{<t}, x) \]

- Use an inference/decoding algorithm to generate text

\[ \hat{y} = \text{decode}(p_\theta(\cdot | x)) \]
- e.g. sampling, \( y_t \sim p_\theta(y_t | y_{<t}, x) \)
- e.g. maximization \( y_t = \arg \max_{y_t} p_\theta(y_t | y_{<t}, x) \)

What is the mass of Jupiter?
Constraints through inference
Model + decoding

• Control: constraints on the generation distribution

Which has the most mass: Mercury, Venus, or Jupiter?
Constraints through inference
Model + decoding

• Control: constraints on the generation distribution
• Goal: decoding algorithms that enable constraints
  • \( \hat{y} = \text{decode}(p_\theta(\cdot | x), \text{constraints}) \)
• Underlying model remains unchanged!

Which has the most mass: Mercury, Venus, or Jupiter?
Constraints through inference
Model + decoding

- Control: *constraints* on the generation distribution
- Goal: *decoding* algorithms that enable constraints
  - \( \hat{y} = \text{decode}(p_{\theta}(\cdot | x), \text{constraints}) \)
  - Underlying model remains unchanged!

- Which *classes* of constraints?
- How to specify and enforce them?

Which has the most mass: Mercury, Venus, or Jupiter?
Constrained generation through inference

• Today: decoding algorithms for constrained generation from two perspectives
Constrained generation through inference

• Today: decoding algorithms for constrained generation from two perspectives

• **Logical lexical constraints** enforced through **discrete inference**

\[(\text{mass} \lor \text{masses}) \land (\text{Mercury}) \land (\text{Venus}) \land (\text{Jupiter})\]
Constrained generation through inference

- Today: decoding algorithms for constrained generation from two perspectives
  - **Logical lexical constraints** enforced through discrete inference
  
    \[(\text{mass } \lor \text{masses}) \land (\text{Mercury}) \land (\text{Venus}) \land (\text{Jupiter})\]

  - **Differentiable constraints** enforced through continuous inference

  
  Which has the most mass: Mercury, Venus, or Jupiter?

  My favorite food is pizza.

  Cats and zebras are my favorite animals.
Constrained generation through *discrete* inference

NeuroLogic A*esque Decoding: Constrained Text Generation with Lookahead Heuristics

In Submission, arxiv:2112.08726

Ximing Lu  Sean Welleck  Peter West  Liwei Jiang  Lianhui Qin  Youngjae Yu  Yejin Choi
Daniel Khashabi  Jungo Kasai  Ronan Le Bras  Rowan Zellers  Noah Smith

UNIVERSITY of WASHINGTON
Logical lexical constraints

- Ensure certain words appear or do not appear

Generate a sentence using cat and fish, but not dog

The cat jumped on the table and saw a fish.
Decoding Objective

Goal: \( y_\star = \arg \max_{y \in \mathcal{Y}} \ log p_\theta(y) + C(y) \)

- \( C(y) \) constraints
- Logical Constraints
  \((\text{cat} \lor \text{cats}) \land (\text{fish}) \land (\neg \text{dog})\)
Standard decoding
Beam search
Standard decoding

Beam search

\[ y_\ast \approx \arg \max_{y \in \mathcal{Y}} \log p_{\theta}(y) + \underbrace{0}_{\text{fluency}} \underbrace{\text{constraints}}_{\text{constraints}} \]
Standard decoding

Beam search

\[ y^* \approx \arg \max_{y \in \mathcal{Y}} \log p_\theta(y) + 0 \]

- Left-to-right search on the lattice of tokens:
Standard decoding

Beam search

\[ y_\ast \approx \arg \max_{y \in \mathcal{Y}} \log p_\theta(y) + 0 \]

- Left-to-right search on the lattice of tokens:
  - Expand prefixes with next-tokens
Standard decoding

Beam search

\[ y_* \approx \arg \max_{y \in \mathcal{Y}} \log p_\theta(y) + 0 \]

- Fluency + Constraints

- Left-to-right search on the lattice of tokens:
  - Expand prefixes with next-tokens
  - Score each using \( \log p_\theta(y_t | y_{<t}) \)

Score each using \( \log p_\theta(y_t | y_{<t}) \)
Standard decoding

Beam search

\[ y_* \approx \arg \max_{y \in \mathcal{Y}} \log p_\theta(y) + \underbrace{0}_{\text{fluency}} \]

- Left-to-right search on the lattice of tokens:
  - Expand prefixes with next-tokens
  - Score each using \( \log p_\theta(y_t | y_{<t}) \)
  - Select the \( k \) best, and repeat
Standard decoding
Beam search

\[ y_* \approx \arg \max_{y \in \mathcal{Y}} \log p_\theta(y) + \text{fluency}^{\sim} 0 \text{ constraints} \]

- Left-to-right search on the lattice of tokens:
  - Expand prefixes with next-tokens
  - Score each using \( \log p_\theta(y_t | y_{<t}) \) fluency
  - Select the \( k \) best, and repeat

=> my cup of water is cold.
Standard decoding

Beam search

\[ y_\ast \approx \arg \max_{y \in \mathcal{Y}} \log p_\theta(y) + \underbrace{0}_{\text{fluency}} \]

- Left-to-right search on the lattice of tokens:
  - Expand prefixes with next-tokens
  - Score each using \( \log p_\theta(y_t | y_{<t}) \)
  - Select the \( k \) best, and repeat

\[ \Rightarrow \text{my cup of water is cold.} \]
Standard decoding
Beam search

\[ y_\ast \approx \arg \max_{y \in \mathcal{Y}} \log p_\theta(y) + \alpha \log p_\theta(y) \]

- Left-to-right search on the lattice of tokens:
  - Expand prefixes with next-tokens
  - Score each using \[ \log p_\theta(y_t | y_{<t}) \]
  - Select the \( k \) best, and repeat

Ignores constraints
Myopic

=> my cup of water is cold.
NeuroLogic decoding [Lu et al 2021]

• + favor tokens that [partially] satisfy constraints

Logical Constraints
\((\text{cat} \vee \text{cats}) \land \text{fish})\)
NeuroLogic decoding [Lu et al 2021]

• + favor tokens that [partially] satisfy constraints

• Keep track of remaining constraints

Logical Constraints
\((\text{cat} \lor \text{cats}) \land \text{fish})\)
NeuroLogic decoding [Lu et al 2021]

- + favor tokens that [partially] satisfy constraints
- Keep track of remaining constraints

Not trivial! Details & other features out of scope for this talk
**NeuroLogic decoding** [Lu et al 2021]

- + favor tokens that [partially] satisfy constraints
- Keep track of remaining constraints
- Score next-tokens using

\[
\log p_\theta(y_t | y_{<t}) + \lambda \max_{c \in \text{remaining}} c(y_t)
\]

Not trivial!

Details & other features out of scope for this talk
NeuroLogic decoding [Lu et al 2021]

- Favor tokens that [partially] satisfy constraints
- Keep track of remaining constraints
- Score next-tokens using

$$ \log p_\theta(y_t | y_{<t}) + \lambda \max_{c \in \text{remaining}} c(y_t) $$

Fluency + Constraints

Logical Constraints

Logical Constraints: $(\text{cat} \lor \text{cats}) \land (\text{fish})$

Not trivial! Details & other features out of scope for this talk

=> my cat is cool.
NeuroLogic decoding [Lu et al 2021]

• + favor tokens that [partially] satisfy constraints

• Keep track of remaining constraints

• Score next-tokens using

\[
\log p_\theta(y_t | y_{<t}) + \lambda \max_{c \in \text{remaining}} c(y_t)
\]

Not trivial! Details & other features out of scope for this talk

Logical Constraints
\((\text{cat} \lor \text{cats}) \land \text{fish}\)

\[
\Rightarrow \text{my cat is cool.}
\]
NeuroLogic A*esque decoding

• Ideally, we want to select next-token candidates on optimal trajectories:

$$\text{argtopk}_{y_t} \left( \max_{y_{>t}} F(y_{<t}, y_t, y_{>t}) \right), \ F = \text{fluency} + \text{constraints}$$
NeuroLogic A*esque decoding

• Ideally, we want to select next-token candidates on optimal trajectories:

\[
\text{argtopk}_{y_t} \left( \max_{y_{>t}} F(y_{<t}, y_t, y_{>t}) \right), \quad F = \text{fluency + constraints}
\]
NeuroLogic A*-esque decoding

- Ideally, we want to select next-token candidates on optimal trajectories:
  \[
  \text{argtopk}_{y_t} \left( \max_{y_{>t}} F(y_{<t}, y_t, y_{>t}) \right), \ F = \text{fluency + constraints}
  \]

- A* Search: best-first search with future heuristics
  \[
  f(a) = s(a) + h(a)
  \]
  score so-far future heuristic
NeuroLogic A*esque decoding

• Approximate with a lookahead heuristic:

\[ \text{argtopk}_y \left( s(y \leq t) + \right) \]

Fluency + constraints-so-far

Logical Constraints
\[(\text{cat} \lor \text{cats}) \land (\text{fish})\]
Approximate with a *lookahead heuristic*:

\[ \text{argtopk}_Y \left( s(y \leq t) + h(y_{<t+\ell}) \right) \]

- **Fluency + constraints-so-far**
- **Probability of satisfying Future constraints**

**Logical Constraints**

\[(\text{cat} \lor \text{cats}) \land (\text{fish})\]

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>cat</th>
<th>cup</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.00</td>
<td>0.25</td>
<td>0.30</td>
<td>0.00</td>
</tr>
</tbody>
</table>
NeuroLogic A*-esque decoding

- Approximate with a lookahead heuristic:

  \[ \text{argtopk}_y t \left( s(y \leq t) + \max \text{ Lookaheads} \right) \]

  Fluency + constraints-so-far

  E.g. single greedy lookahead

  Probability of satisfying Future constraints

  \[ h(y < t + \ell) \]

Logical Constraints

(cat \lor cats) \land (fish)

Fluency + constraints-so-far

E.g. single greedy lookahead

Probability of satisfying Future constraints
**NeuroLogic A*esque decoding**

- Approximate with a *lookahead heuristic*:

  \[ \text{argtopk}_y(t) \left( s(y \leq t) + \max \text{ Lookaheads} h(y < t + \ell) \right) \]

  - Fluency + constraints-so-far
  - E.g. single greedy lookahead
  - Probability of satisfying Future constraints

**Logical Constraints**

\((cat \lor cats) \land (fish)\)
NeuroLogic A*esque decoding

- Approximate with a lookahead heuristic:

\[ \text{argtopk}_y \left( s(y \leq t) + \max \text{ Lookaheads} \right) \]

Fluency + constraints-so-far

E.g. single greedy lookahead

Probability of satisfying Future constraints

Logical Constraints

\((\text{cat} \lor \text{cats}) \land (\text{fish})\)

=> my \text{cup} has a \text{fish} and \text{cat} on it.
NeuroLogic A*esque decoding

• Approximate with a lookahead heuristic:

\[
\text{argtopk}_t \left( \sum_{y \leq t} s(y) + \max \text{ Lookaheads} \right),
\]

\[
\text{Probability of satisfying Future constraints}
\]

\[
\text{Fluency + constraints-so-far}
\]

\[
\text{E.g. single greedy lookahead}
\]

Logical Constraints

\[(\text{cat} \lor \text{cats}) \land (\text{fish})\]

=> my \text{cup} has a \text{fish} and \text{cat} on it.
NeuroLogic A*esque decoding

- Approximate with a lookahead heuristic:
  \[ \text{argtopk}_y(t) \left( s(y \leq t) + \max \text{ Lookaheads} \ h(y < t + \ell) \right) \]

- Fluency + constraints-so-far
- E.g. single greedy lookahead
- Probability of satisfying Future constraints

- “A*esque”: beam instead of best-first

Logical Constraints
\[(\text{cat} \lor \text{cats}) \land (\text{fish})\]

E.g. single greedy lookahead

Probability of satisfying Future constraints

=> my cup has a fish and cat on it.
CommonGen
(Lin et al., 2020)

- Standard constrained generation benchmark:
  ~60k train, ~7k test

  Constraints: {sponge, pour, pool, side, clean}
  Example output: Pour water on a sponge and use it to clean the side of the pool.
CommonGen
(Lin et al., 2020)

- Standard constrained generation benchmark:
  ~60k train, ~7k test

  Constraints: {sponge, pour, pool, side, clean}
  Example output: Pour water on a sponge and use it to clean the side of the pool.

beam search

The woman, whose name has not been released, was taken to a local hospital, where she was listed in stable condition, according to the sheriff's office.

completely irrelevant
CommonGen
(Lin et al., 2020)

- Standard constrained generation benchmark:
  ~60k train, ~7k test

  Constraints: \{sponge, pour, pool, side, clean\}
  Example output: Pour water on a sponge and use it to clean the side of the pool.

beam search

The woman, whose name has not been released, was taken to a local hospital, where she was listed in stable condition, according to the sheriff's office.

completely irrelevant

NeuroLogic
The man cleans a sponge in a pouring pool at the side of the road.

slightly awkward

(C (sponge ∨ sponges) ∧ (pour ∨ pours ∨ pouring ∨ poured) ∧ (pool ∨ pools) ∧ (side ∨ sides) ∧ (clean ∨ clean ∨ cleans ∨ cleaning))
**CommonGen**  
(Lin et al., 2020)

- Standard constrained generation benchmark:  
  ~60k train, ~7k test

  Constraints: \{sponge, pour, pool, side, clean\}
  Example output: Pour water on a sponge and use it to clean the side of the pool.

  beam search

  The woman, whose name has not been released, was taken to a local hospital, where she was listed in stable condition, according to the sheriff's office.

  completely irrelevant

  The man cleans a sponge in a pouring pool at the side of the road.

  slightly awkward

  The boy cleaned the side of the pool with a sponge, and poured water over it.
<table>
<thead>
<tr>
<th>Human evaluation</th>
<th>CommonGen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-tuned GPT-2</td>
<td>(Lin et al., 2020)</td>
</tr>
<tr>
<td></td>
<td>Off-the-shelf GPT-2</td>
</tr>
</tbody>
</table>
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

- CBS
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Off-the-shelf GPT-2

Quality

2.8
2.6
2.5
2.1
1.8
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

- CBS
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Off-the-shelf GPT-2

Quality

2.27
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

<table>
<thead>
<tr>
<th>Method</th>
<th>CBS</th>
<th>NeuroLogic A*esq (greedy)</th>
<th>NeuroLogic A*esq (beam)</th>
<th>NeuroLogic A*esq (sample)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBS</td>
<td>2.27</td>
<td>2.54</td>
<td>2.27</td>
<td>2.27</td>
</tr>
<tr>
<td>Off-the-shelf GPT-2</td>
<td>2.54</td>
<td>2.34</td>
<td>2.27</td>
<td>2.27</td>
</tr>
</tbody>
</table>
Human evaluation | CommonGen

Fine-tuned GPT-2

- CBS
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Off-the-shelf GPT-2

Quality

2.8
2.6
2.4
2.2
2.0
1.8

2.27
2.34
2.66
Human evaluation | CommonGen

Fine-tuned GPT-2

- CBS
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Off-the-shelf GPT-2

(Lin et al., 2020)
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

- CBS
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Off-the-shelf GPT-2

<table>
<thead>
<tr>
<th></th>
<th>CBS</th>
<th>NeuroLogic</th>
<th>NeuroLogic A*esq (greedy)</th>
<th>NeuroLogic A*esq (beam)</th>
<th>NeuroLogic A*esq (sample)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.27</td>
<td>2.54</td>
<td>2.66</td>
<td>2.68</td>
<td>2.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.54</td>
<td>2.66</td>
<td>2.68</td>
<td>2.66</td>
</tr>
</tbody>
</table>
Human evaluation | CommonGen
(Lin et al., 2020)

**Fine-tuned GPT-2**
- CBS
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

**Off-the-shelf GPT-2**
- TSMH
- NeuroLogic
- NeuroLogic A*esq (greedy)
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

- CBS
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Off-the-shelf GPT-2

- TSMH
- NeuroLogic
- NeuroLogic A*esq (greedy)
Human evaluation | CommonGen

Fine-tuned GPT-2

- CBS
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Off-the-shelf GPT-2

- TSMH
- NeuroLogic
- NeuroLogic A*esq (greedy)
Human evaluation | CommonGen

Fine-tuned GPT-2

- CBS
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Off-the-shelf GPT-2

- TSMH
- NeuroLogic
- NeuroLogic A*esq (greedy)
Human evaluation | CommonGen
(Lin et al., 2020)

Fine-tuned GPT-2
- CBS
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Off-the-shelf GPT-2
- TSMH
- NeuroLogic
- Off-the-shelf A*esq (greedy)

Off-the-shelf A* outperforms all fine-tuned methods
### Human evaluation

**CommonGen**

(Lin et al., 2020)

**Fine-tuned GPT-2**
- CBS
- NeuroLogic
- NeuroLogic $A^*$esq (greedy)
- NeuroLogic $A^*$esq (beam)
- NeuroLogic $A^*$esq (sample)

<table>
<thead>
<tr>
<th>Method</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBS</td>
<td>2.27</td>
</tr>
<tr>
<td>NeuroLogic</td>
<td>2.54</td>
</tr>
<tr>
<td>NeuroLogic $A^*$esq (greedy)</td>
<td>2.66</td>
</tr>
<tr>
<td>NeuroLogic $A^*$esq (beam)</td>
<td>2.68</td>
</tr>
<tr>
<td>NeuroLogic $A^*$esq (sample)</td>
<td>2.66</td>
</tr>
</tbody>
</table>

**A* NeuroLogic with greedy lookahead: efficient & performant**

**Off-the-shelf GPT-2**
- TSMH
- Off-the-shelf A* (greedy)

<table>
<thead>
<tr>
<th>Method</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSMH</td>
<td>1.85</td>
</tr>
<tr>
<td>Off-the-shelf A* (greedy)</td>
<td>2.64</td>
</tr>
</tbody>
</table>

**Off-the-shelf A* outperforms all fine-tuned methods**
Enables many constrained generation tasks
Enables many constrained generation tasks

Constrained MT
(Dinu et al., 2019)

- MarianMT
- Post and Vilar (2018)
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

BLEU

53.8
53.6
53.5
53.3
53.1
52.8

52.9 53 33.4 33.7 33.7 33.6
Enables many constrained generation tasks

Constrained MT
(Dinu et al., 2019)

- MarianMT
- Post and Vilar (2018)
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Few-Shot E2ENLG
(Chen et al., 2020)

- KGPT-Graph
- KGPT-Seq
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)
Enables many constrained generation tasks

Constrained MT
(Dinu et al., 2019)

- MarianMT
- Post and Vilar (2018)
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Few-Shot E2ENLG
(Chen et al., 2020)

- KGPT-Graph
- KGPT-Seq
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)

Question Generation
(Zhang et al., 2020)

- CGMH
- TSMH
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)
• Greedy lookahead length (CommonGen)
• Improves at varying amounts of training data

Figure 3: Performance (y-axis) of supervised GPT-2 on E2ENLG, with a varying amount of training data for supervision (x-axis). The purple, blue, and black line denote decoding with NEUROLOGIC★, NEUROLOGIC and conventional beam search respectively.
Constrained generation through *discrete* inference

**A* Neurologic**

- **Constraints**: expressive class of lexical constraints
- **Search**: discrete with future approximation
- **Enables**: constraints without fine-tuning, better fine-tuned performance

NeuroLogic A*esque Decoding:
Constrained Text Generation with Lookahead Heuristics

[arxiv:2112.08726](https://arxiv.org/abs/2112.08726)
[github.com/GloriaXimingLu/star_neurologic](https://github.com/GloriaXimingLu/star_neurologic)
Constrained generation through inference

- Today: algorithms for constrained generation from two perspectives
  - **Logical lexical constraints** enforced through **discrete inference**

\[
(mass \lor masses) \land (Mercury) \land (Venus) \land (Jupiter)
\]

- *Which has the most mass: Mercury, Venus, or Jupiter?*

- **Differentiable constraints** enforced through **continuous inference**

- Language Model $f_{\text{fluency}}$
  - My favorite food is pizza.

- Similarity $f_{\text{similarity}}$
  - Cats and zebras are my favorite animals.

- Keywords $f_{\text{keywords}}$
  - cat, zebra
Constrained generation through continuous inference

COLD Decoding:
Constrained Decoding with Langevin Dynamics

In Submission, arxiv:2202.11705

Lianhui Qin, Sean Welleck, Daniel Khashabi, Yejin Choi
Lexically Constrained Generation

Keywords

\{ \text{mass, Mercury, Jupiter} \}

Generation

\text{Jupiter} has more \text{mass} than \text{Mercury}. 
Lexically Constrained Generation

Keywords: {mass, Mercury, Jupiter}

Constraints: $f_{\text{fluency}}(y)$

Generation: Jupiter has more mass than Mercury.
Lexically Constrained Generation

Keywords: \{ mass, Mercury, Jupiter \}

Generation:
Jupiter has more mass than Mercury.

Constraints:
- Fluency constraint: \( f_{fluency}(y) \)
- Task-specific constraint: \( f_{keywords}(y) \)

- Language Model
Text infilling / abductive reasoning

*Left context*

She went to practice everyday.
Text infilling / abductive reasoning

Left context
She went to practice everyday.

Right context
She won a gold medal in the Olympic marathon.
Text infilling / abductive reasoning

**Left context**
She went to practice everyday.

**Generation**
She ran a lot of miles at practice.

**Right context**
She won a gold medal in the Olympic marathon.

AbductiveNLG
(Bhagavatula et al., 2020)
She went to practice everyday.

She ran a lot of miles at practice.

She won a gold medal in the Olympic marathon.

Constraints:

Language Model $f_{fluency}(y)$

Fluency constraint
She went to practice everyday.

She ran a lot of miles at practice.

She went to practice everyday.

She won a gold medal in the Olympic marathon.

Constraints:
- Fluency constraint: $f_{fluency}(y)$
- Task-specific constraints: $f_{coherence-left}(y)$

AbductiveNLG (Bhagavatula et al., 2020)
Text infilling / abductive reasoning

Constraints:

Language Model: $f_{\text{fluency}}(y)$

She went to practice everyday.

Fluency constraint

She ran a lot of miles at practice.

Task-specific constraints

She won a gold medal in the Olympic marathon.

She went to practice...

$f_{\text{coherence-left}}(y)$

$f_{\text{coherence-right}}(y)$
Text similarity / counterfactual reasoning

The law student joined a prestigious law firm after graduating.

(Quin et al., 2019)
Text similarity / counterfactual reasoning

The law student

joined a prestigious law firm after graduating.

The medical student

TimeTravel

(Qin et al., 2019)

Keep Similar
Text similarity / counterfactual reasoning

The law student

joined a prestigious law firm after graduating.

The medical student

joined a prestigious medical practice after graduation.

TimeTravel
(Qin et al., 2019)

Keep Similar

Generation
Text similarity / counterfactual reasoning

The law student joined a prestigious law firm after graduating.

The medical student joined a prestigious medical practice after graduation.

Constraints: Fluency constraint

Language Model $f_{\text{fluency}}(Y)$

TimeTravel (Qin et al., 2019)
Text similarity / counterfactual reasoning

The law student

joined a prestigious law firm after graduating.

The medical student

joined a prestigious medical practice after graduation.

TimeTravel (Qin et al., 2019)

Constraints:

Language Model

$\text{fluency}(y)$

The medical student

$\text{coherence-left}(y)$

Fluency constraint

Task-specific constraints
Text similarity / counterfactual reasoning

The law student joined a prestigious law firm after graduating.

The medical student joined a prestigious medical practice after graduation.

Constraints:

- Fluency constraint: \( f_{\text{fluency}}(y) \)
- Task-specific constraints: \( f_{\text{coherence-left}}(y) \), \( f_{\text{similarity}}(y, y^*) \)

TimeTravel
(Qin et al., 2019)
Constrained generation as sampling from an energy-based model

Energy function:

\[ E(y) = f_{\text{fluency}}(y) + f_1(y) + f_2(y) + \ldots \]
Constrained generation as sampling from an energy-based model

Fluency constraint

Energy function:
\[ E(y) = f_{\text{fluency}}(y) + f_1(y) + f_2(y) + \ldots \]

Energy-based model:
\[ p(y) = \exp \left\{ -E(y) \right\} / Z \]

Task-specific constraints
Constrained generation as sampling from an energy-based model

Energy function:
\[ E(y) = f_{\text{fluency}}(y) + f_1(y) + f_2(y) + \ldots \]

Energy-based model:
\[ p(y) = \exp \left\{ -E(y) \right\} / Z \]

Constrained generation:
\[ \hat{y} \sim p(y) \]
Sampling from an energy-based model

Constrained generation: \( \hat{y} \sim \exp \{-E(y)\}/Z \)

- Gradient-free MCMC (e.g. Gibbs sampling [Bishop & Nasrabadi 2006]): slow
Sampling from an energy-based model

Constrained generation: \( \hat{y} \sim \exp\{-E(y)\}/Z \)

- Gradient based MCMC, e.g. Langevin dynamics [Welling & Teh, 2011; Du & Mordatch, 2019]

\[
\tilde{y}^{(n)} = \tilde{y}^{(n-1)} - \eta \nabla_{\tilde{y}} E(\tilde{y}) + \epsilon \quad \epsilon \sim N(0,1)
\]
Sampling from an energy-based model

Constrained generation: $\hat{y} \sim \exp \{-E(y)\}/Z$

• Gradient based MCMC, e.g. Langevin dynamics [Welling & Teh, 2011; Du & Mordatch, 2019]

$$\tilde{y}^{(n)} = \tilde{y}^{(n-1)} - \eta \nabla_{\tilde{y}} E(\tilde{y}) + \epsilon \quad \epsilon \sim N(0,1)$$

More efficient sampling by using the gradient of $E(\tilde{y})$
Sampling from an energy-based model

Constrained generation: \( \hat{y} \sim \exp \{ -E(y) \} / Z \)

- Gradient based MCMC, e.g. Langevin dynamics [Welling & Teh, 2011; Du & Mordatch, 2019]

\[
\tilde{y}^{(n)} = \tilde{y}^{(n-1)} - \eta \nabla_{\tilde{y}} E(\tilde{y}) + \epsilon \quad \epsilon \sim N(0,1)
\]

More efficient sampling by using the gradient of \( E(\tilde{y}) \)

\( \nabla_{y} E(y) \) not defined for discrete \( y \)
Sampling from an energy-based model

**Constrained generation:** \( \hat{y} \sim \exp \{-E(y)\} / Z \)

- Define energy over "soft sequence" of continuous vectors:
  - \( \tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_T) \), where \( \tilde{y}_t \in \mathbb{R}^{vocab} \)

<table>
<thead>
<tr>
<th>vocab</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>4.1</td>
</tr>
<tr>
<td>1.9</td>
<td>0.9</td>
</tr>
<tr>
<td>-5.</td>
<td>2.2</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Sampling from an energy-based model

Constrained generation: \[ \hat{y} \sim \exp \{ -E(y) \} / Z \]

- Define energy over "soft sequence" of continuous vectors:
  - \( \tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_T) \), where \( \tilde{y}_t \in \mathbb{R}^{\text{vocab}} \)

  \[
  \begin{pmatrix}
  0.1 & 4.1 & 0.7 \\
  1.9 & 0.9 & 3.1 \\
  -5. & 2.2 & -3. \\
  0.2 & 0.2 & 1.1 \\
  \end{pmatrix}
  \] 

  \[
  \begin{pmatrix}
  0.1 \\
  99. \\
  0.1 \\
  0.1 \\
  \end{pmatrix}
  \] 

  \[ \text{softmax}(\cdot) \longrightarrow \begin{pmatrix}
  0 \\
  1 \\
  0 \\
  0 \\
  \end{pmatrix} \]

  \[ \text{dog} \]

- Discrete token: \( \text{softmax}(\tilde{y}_t / \tau) \) as \( \tau \to 0 \)
Sampling from an energy-based model

Constrained generation: \( \hat{y} \sim \exp \{-E(y)\}/Z \)

- Constraints as **differentiable functions**

\[
f_{fluency}(\tilde{y})
\]

\[
f_{LM}(\tilde{y}) = \sum_{t=1}^{T} \sum_{v \in \mathcal{V}} p_{LM}(v|\tilde{y}_{<t}) \log \text{softmax}(\hat{y}_t(v))
\]
Sampling from an energy-based model

Constrained generation: \( \hat{y} \sim \exp \left\{ -E(y) \right\} / Z \)

- Constraints as **differentiable functions**

\[
f_{\text{keywords}}(\tilde{y}) = \sum \text{Mass, Jupiter, Mercury}
\]

\[
f_{\text{similarity}}(\tilde{y}, y_*) = \text{ngram-match}(\tilde{y}, y_*)
\]

\( f_{\text{sim}}(\tilde{y}; y_*) = \text{ngram-match}(\tilde{y}, y_*) \)  \quad \text{(Liu et al., 2021)}
Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:
Specify energy $E(\tilde{y}) = \sum_{i} f_i(\tilde{y})$, then:

$$
\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla \tilde{y} E(\tilde{y}^{(n)}) + \epsilon^{(n)}
$$

Initial distribution

Langevin Dynamics

Target constrained distribution
Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization

Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization

Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization

Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization

Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization

Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization

Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization

Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization

Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization

Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization

Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

Langevin Dynamics

$$\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}$$

Initial distribution

Target constrained distribution

Discretization
Specify energy $E(\tilde{y}) = \sum_i f_i(\tilde{y})$, then:

\[
\tilde{y}^{(n+1)} \leftarrow \tilde{y}^{(n)} - \eta \nabla_{\tilde{y}} E(\tilde{y}^{(n)}) + \epsilon^{(n)}
\]

**Initial distribution**

**Target constrained distribution**

**Discretization**

Apply directly to **off-the-shelf** left-to-right language models **without** the need for any task-specific fine-tuning.
Lexically constrained generation

We specify an energy function of the following form:

$$E(\tilde{y}) = \lambda_a l_{tr} f_{LM}(\tilde{y}) + \lambda_a r_{tr} f_{LM}(\tilde{y}) + \lambda_b f_{sim}(\tilde{y}; W) + \lambda_c f_{prea}(\tilde{y}; c(W)).$$

| Models          | Coverage |     | Fluency |
|-----------------|----------|-----|
|                 | Count    | Percent | PPL     | Human |
| TSMH            | 2.72     | 71.27 | 1545.15 | 1.72  |
| NEUROLOGIC      | 3.30     | 91.00 | 28.61   | 2.53  |
| COLD (ours)     | 4.24     | 94.50 | 54.98   | 2.07  |
Lexically constrained generation

We specify an energy function of the following form:

\[
E(\tilde{y}) = \lambda_a^I f_{LM}^I(\tilde{y}) + \lambda_a^R f_{LM}^R(\tilde{y}) + \lambda_b f_{sim}(\tilde{y}; W) + \lambda_c f_{prea}(\tilde{y}; c(W)).
\]

<table>
<thead>
<tr>
<th>Models</th>
<th>Coverage Count</th>
<th>Coverage Percent</th>
<th>Fluency PPL</th>
<th>Fluency Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSMH</td>
<td>2.72</td>
<td>71.27</td>
<td>1545.15</td>
<td>1.72</td>
</tr>
<tr>
<td>NEUROLOGIC</td>
<td>3.30</td>
<td>91.00</td>
<td>28.61</td>
<td>2.53</td>
</tr>
<tr>
<td>COLD (ours)</td>
<td><strong>4.24</strong></td>
<td><strong>94.50</strong></td>
<td>54.98</td>
<td>2.07</td>
</tr>
</tbody>
</table>

- Good constraint coverage

CommonGen

(Lin et al., 2020)
Lexically constrained generation

We specify an energy function of the following form:

\[ E(\tilde{y}) = \lambda_a f_{LM}(\tilde{y}) + \lambda_b f_{sim}(\tilde{y}; W) + \lambda_c f_{pred}(\tilde{y}; c(W)) \]

<table>
<thead>
<tr>
<th>Models</th>
<th>Coverage</th>
<th>Fluency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Count</td>
<td>Percent</td>
</tr>
<tr>
<td>TSMH</td>
<td>2.72</td>
<td>71.27</td>
</tr>
<tr>
<td>NEUROLOGIC</td>
<td>3.30</td>
<td>91.00</td>
</tr>
<tr>
<td>COLD (ours)</td>
<td><strong>4.24</strong></td>
<td><strong>94.50</strong></td>
</tr>
</tbody>
</table>

- Good constraint coverage
- Competitive fluency with lexical-specific NeuroLogic
Abductive reasoning

- Enables **left** and **right** coherence while staying **fluent**

\[
E(\tilde{y}) = \lambda_{a}^{lr} f_{lM}^{\rightarrow}(\tilde{y}; x_l) + \lambda_{a}^{rl} f_{lM}^{\rightarrow}(\tilde{y}; x_r) + \lambda_{b} f_{pred}(\tilde{y}; x_r) + \lambda_{c} f_{sim}(\tilde{y}; kw(x_r) - kw(x_l)).
\]

<table>
<thead>
<tr>
<th>Begin. $x_l$</th>
<th>Tim wanted to learn astronomy.</th>
</tr>
</thead>
<tbody>
<tr>
<td>End. $x_r$</td>
<td>Tim worked hard in school to become one.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LEFT-ONLY</th>
<th>He was a good student.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELOREAN</td>
<td>So he bought a telescope.</td>
</tr>
<tr>
<td>COLD (ours)</td>
<td>He wanted to become a professional astronomer.</td>
</tr>
</tbody>
</table>
Abductive reasoning

- Enables **left** and **right** coherence while staying **fluent**

\[
E(\tilde{y}) = \lambda^l f_{\text{LM}}(\tilde{y}; x_l) + \lambda^r f_{\text{LM}}(\tilde{y}; x_r) + \lambda_{\text{Pred}} f_{\text{Pred}}(\tilde{y}; x_r) + \lambda_{\text{Sim}} f_{\text{Sim}}(\tilde{y}; \text{kw}(x_r) - \text{kw}(x_l)).
\]

<table>
<thead>
<tr>
<th>Models</th>
<th>Automatic Eval</th>
<th>Human Eval</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLEU\textsubscript{4}</td>
<td>ROUGE-L</td>
</tr>
<tr>
<td>LEFT-ONLY</td>
<td>0.88</td>
<td>16.26</td>
</tr>
<tr>
<td>DELOREAN</td>
<td>1.60</td>
<td>19.06</td>
</tr>
<tr>
<td>COLD (ours)</td>
<td><strong>1.79</strong></td>
<td><strong>19.50</strong></td>
</tr>
</tbody>
</table>

AbductiveNLG
(Bhagavatula et al., 2020)
Abductive reasoning

(Bhagavatula et al., 2020)

<table>
<thead>
<tr>
<th>top-k</th>
<th>Grammar</th>
<th>Left-coher. (x-y)</th>
<th>Right-coher. (y-z)</th>
<th>Overall-coher. (x-y-z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4.38</td>
<td><strong>3.99</strong></td>
<td>2.88</td>
<td>2.92</td>
</tr>
<tr>
<td>5</td>
<td>4.27</td>
<td>3.71</td>
<td>3.04</td>
<td>2.87</td>
</tr>
<tr>
<td>10</td>
<td>4.09</td>
<td>3.84</td>
<td><strong>3.09</strong></td>
<td><strong>2.94</strong></td>
</tr>
<tr>
<td>50</td>
<td>3.95</td>
<td>3.62</td>
<td>3.07</td>
<td>2.87</td>
</tr>
<tr>
<td>100</td>
<td>3.80</td>
<td>3.54</td>
<td>3.03</td>
<td>2.84</td>
</tr>
</tbody>
</table>

*Table 6. Ablation for the effect of k in top-k filtering mechanism (§3.3). We use the same setting as Table 5.*

- **Discretization** step important: low fluency with large k
Abductive reasoning

AbductiveNLG
(Bhagavatula et al., 2020)

<table>
<thead>
<tr>
<th>top-(k)</th>
<th>Grammar</th>
<th>Left-coher. (x-y)</th>
<th>Right-coher. (y-z)</th>
<th>Overall-coher. (x-y-z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4.38</td>
<td>3.99</td>
<td>2.88</td>
<td>2.92</td>
</tr>
<tr>
<td>5</td>
<td>4.27</td>
<td>3.71</td>
<td>3.04</td>
<td>2.87</td>
</tr>
<tr>
<td>10</td>
<td>4.09</td>
<td>3.84</td>
<td><strong>3.09</strong></td>
<td><strong>2.94</strong></td>
</tr>
<tr>
<td>50</td>
<td>3.95</td>
<td>3.62</td>
<td>3.07</td>
<td>2.87</td>
</tr>
<tr>
<td>100</td>
<td>3.80</td>
<td>3.54</td>
<td>3.03</td>
<td>2.84</td>
</tr>
</tbody>
</table>

*Table 6. Ablation for the effect of \(k\) in top-\(k\) filtering mechanism (§3.3). We use the same setting as Table 5.*

- **Discretization** step important: low fluency with large \(k\)
- **COLD sampling** important: low right-coherence with small \(k\)
Abductive reasoning

(Bhagavatula et al., 2020)

<table>
<thead>
<tr>
<th>Models</th>
<th>Grammar</th>
<th>Left-coher. (x-y)</th>
<th>Right-coher. (y-z)</th>
<th>Overall-coher. (x-y-z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLD (Full)</td>
<td>4.17</td>
<td>3.96</td>
<td>2.88</td>
<td>2.83</td>
</tr>
<tr>
<td>COLD $- f_{sim}$</td>
<td>4.54</td>
<td>3.82</td>
<td>2.73</td>
<td>2.69</td>
</tr>
<tr>
<td>COLD $- f_{LM}$</td>
<td>4.35</td>
<td>3.97</td>
<td>2.84</td>
<td>2.80</td>
</tr>
<tr>
<td>COLD $- f_{pred}$</td>
<td><strong>4.61</strong></td>
<td><strong>4.07</strong></td>
<td>2.75</td>
<td>2.77</td>
</tr>
</tbody>
</table>

*Table 5. Ablation for the effect of different constraints in Eq. (7).*

We use the abductive reasoning task as a case study, with human evaluation on 125 test examples. The best overall coherence is achieved when all the constraints are present.

- Right-hand constraints are important for right-hand coherence!
Constrained generation through *continuous* inference

- **Constraints:** differentiable constraints; fluency, keywords, similarity
- **Search:** Langevin dynamics + discretization
- **Enables:** constraints without additional fine-tuning

**COLD Decoding:**
Constrained Decoding with Langevin Dynamics

arxiv:2202.11705
github.com/qkaren/COLD_decoding
Constrained generation

Looking ahead
Constrained generation

Looking ahead

- Grounded generation
Constrained generation

Looking ahead

- Grounded generation

NaturalProofs: Mathematical Theorem Proving in Natural Language
Towards Grounded Natural Language Proof Generation (Work in Progress)

- Joint learning & inference
Thanks for your attention!