

Constrained text generation through discrete & continuous inference

Sean Welleck | 03.11.2022

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?

< 175B best-of-64 >

There are many explanations for the exponential growth in technology in the last century. One explanation is that the pace of technological progress speeds up exponentially over time because of a common force that pushes it forward^[3]. Another explanation is that each new generation of technology stands on the shoulders of its predecessors, allowing for innovations that lead to the next generation of even better

Machine Translation

Google Translate

Text Documents Websites

DETECT LANGUAGE ENGLISH SPANISH ENGLISH SPANISH ARABIC

Translation

0 / 5,000

Program Synthesis

Your AI pair programmer

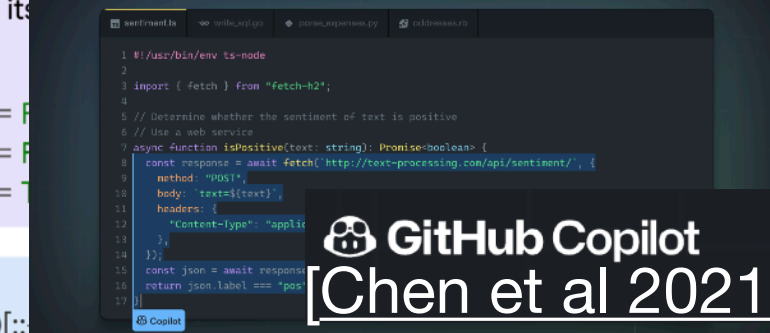
With GitHub Copilot, get suggestions for whole lines or entire functions right inside your editor.

prompt
Write a python function that returns true if a number is one less than twice its value and false otherwise. It should satisfy these tests:

```
assert check(70) == False
assert check(23) == True
assert check(73) == True
```

model
def check(n):
 if n == 2*int(str(n)[::-1]):
 return True
 else:
 return False

[Austin et al 2021]



GitHub Copilot
[Chen et al 2021]

Dialogue

Hi! Hello, I am a friendly dialog model. What do you want to talk about?

👍 🗨️

What's a good topic for a new blog?

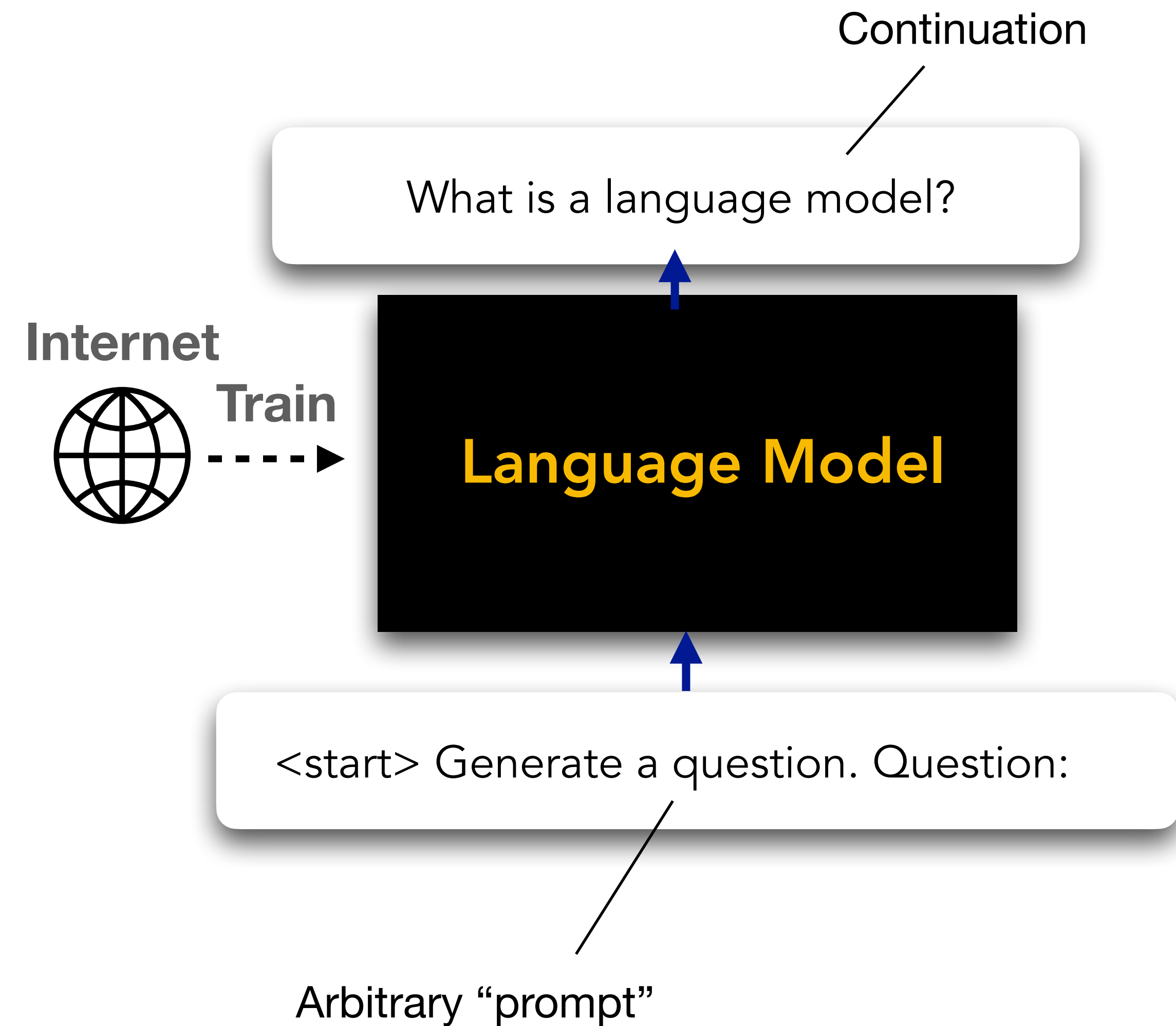
Well there are so many! How about something about a new food item that you just tried.

👍 🗨️

[Thoppilan et al 2022]

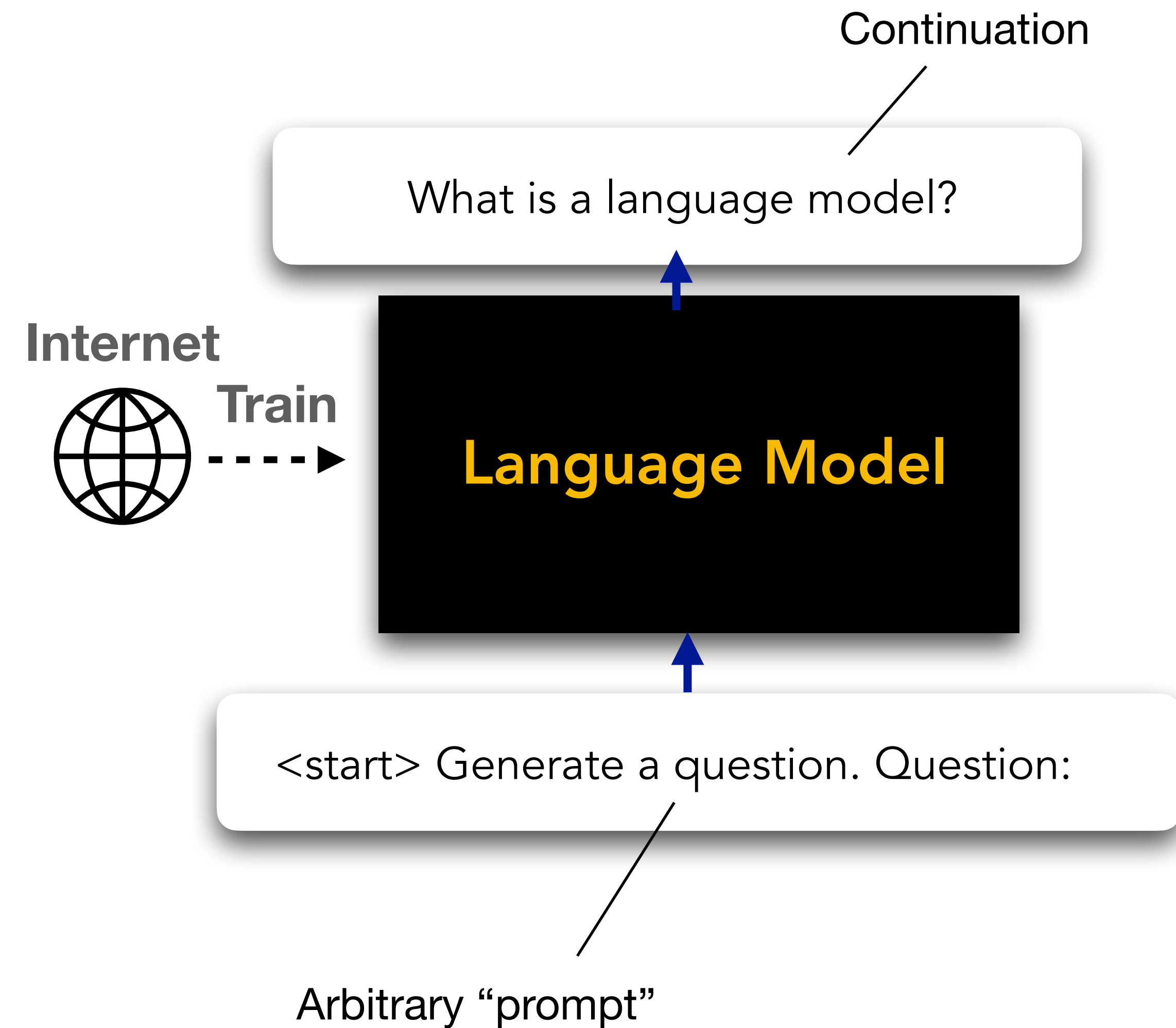
Neural text generation

- **General purpose:**

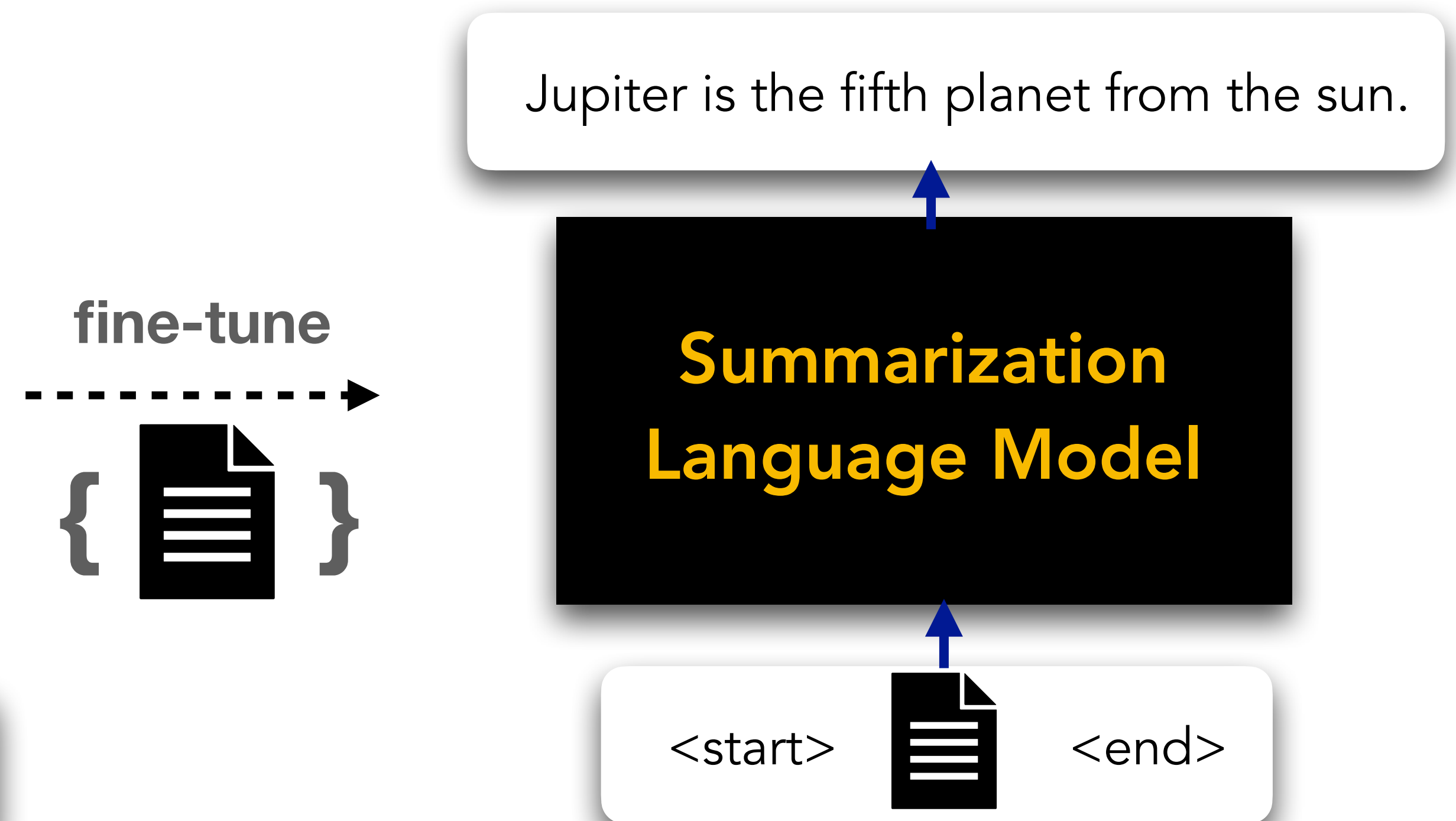


Neural text generation

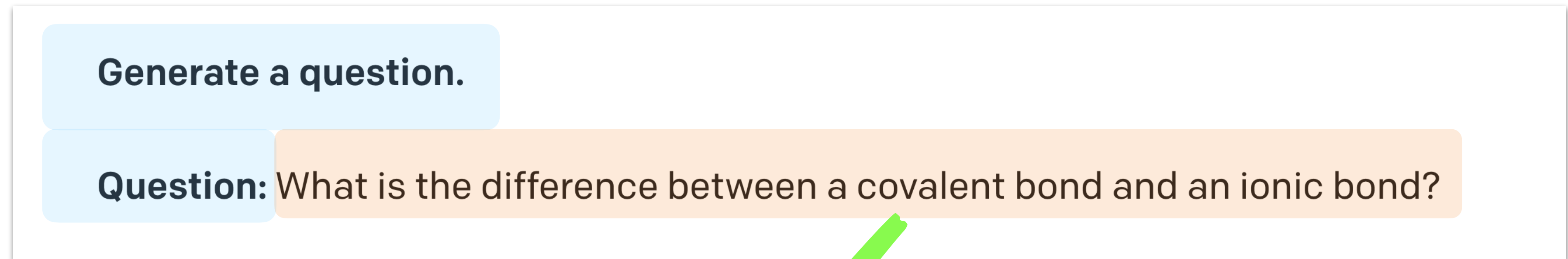
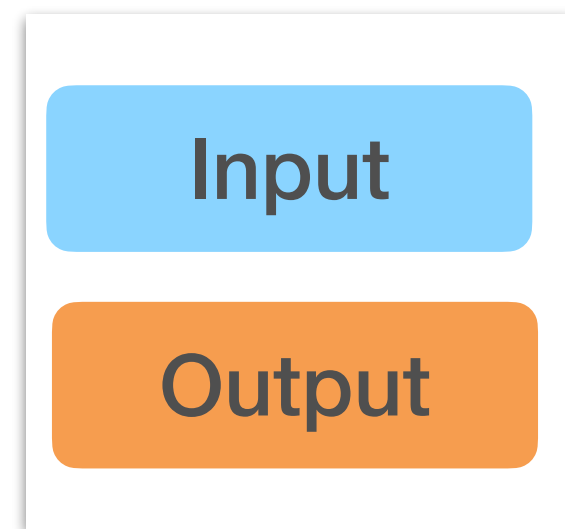
- **General purpose:**



- **Task-specific:**



- **GPT-3:** a *general purpose* 175B parameter language model:

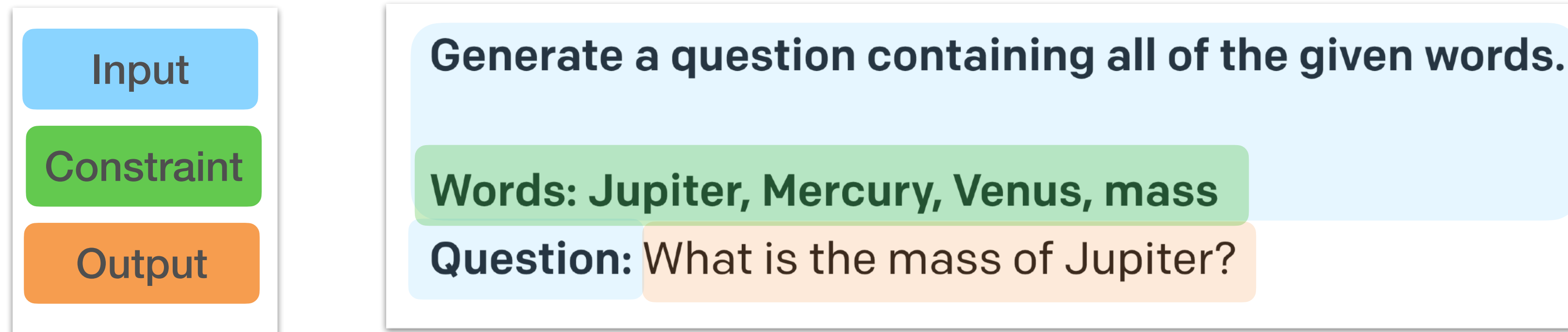


- **GPT-3**: a *general purpose* 175B parameter language model:

<p>Input</p>	<p>Summarize this for a second-grade student:</p> <h2>Jupiter</h2> <p>From Wikipedia, the free encyclopedia</p> <p><i>This article is about the planet. For the Roman god, see Jupiter (mythology). For other uses, see Jupiter (disambiguation).</i></p> <p>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 size.</p>
<p>Output</p>	<p>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.</p>

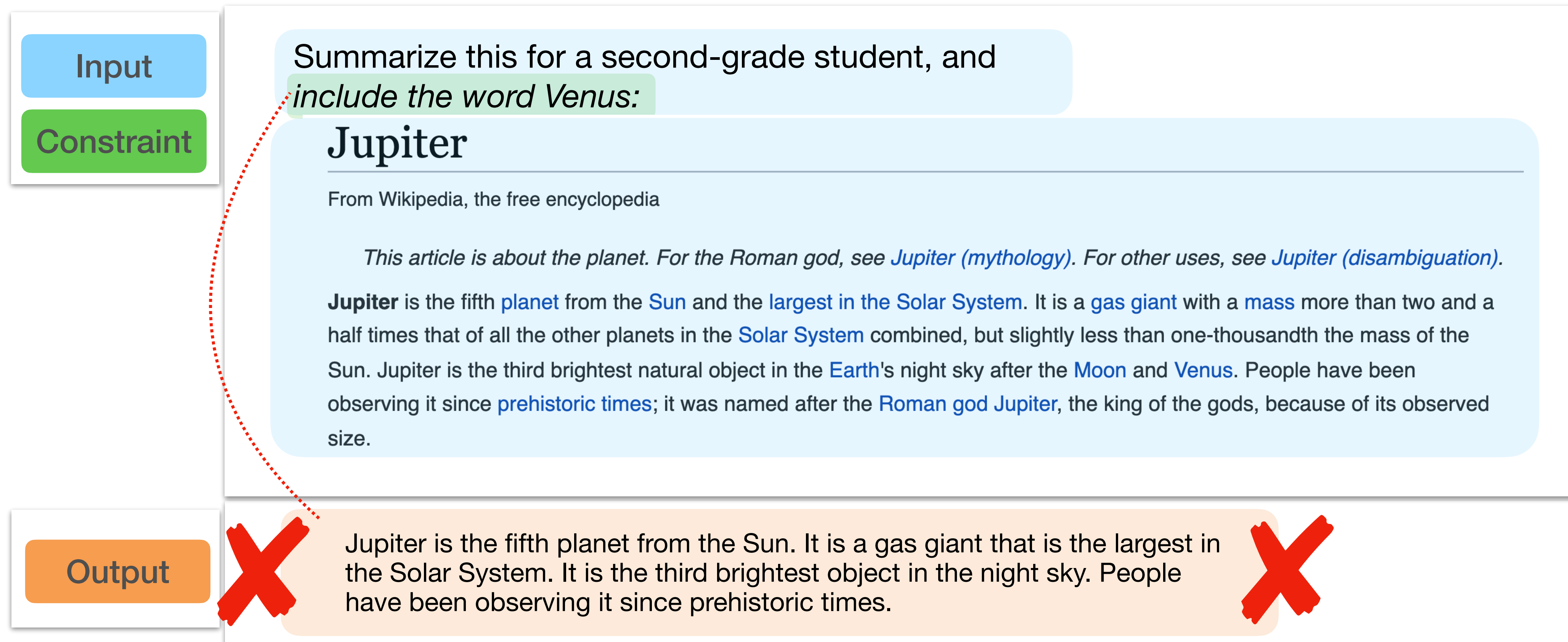
Controlling neural text generation

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



Controlling neural text generation

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



Controlling neural text generation

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

Input

Constraint

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

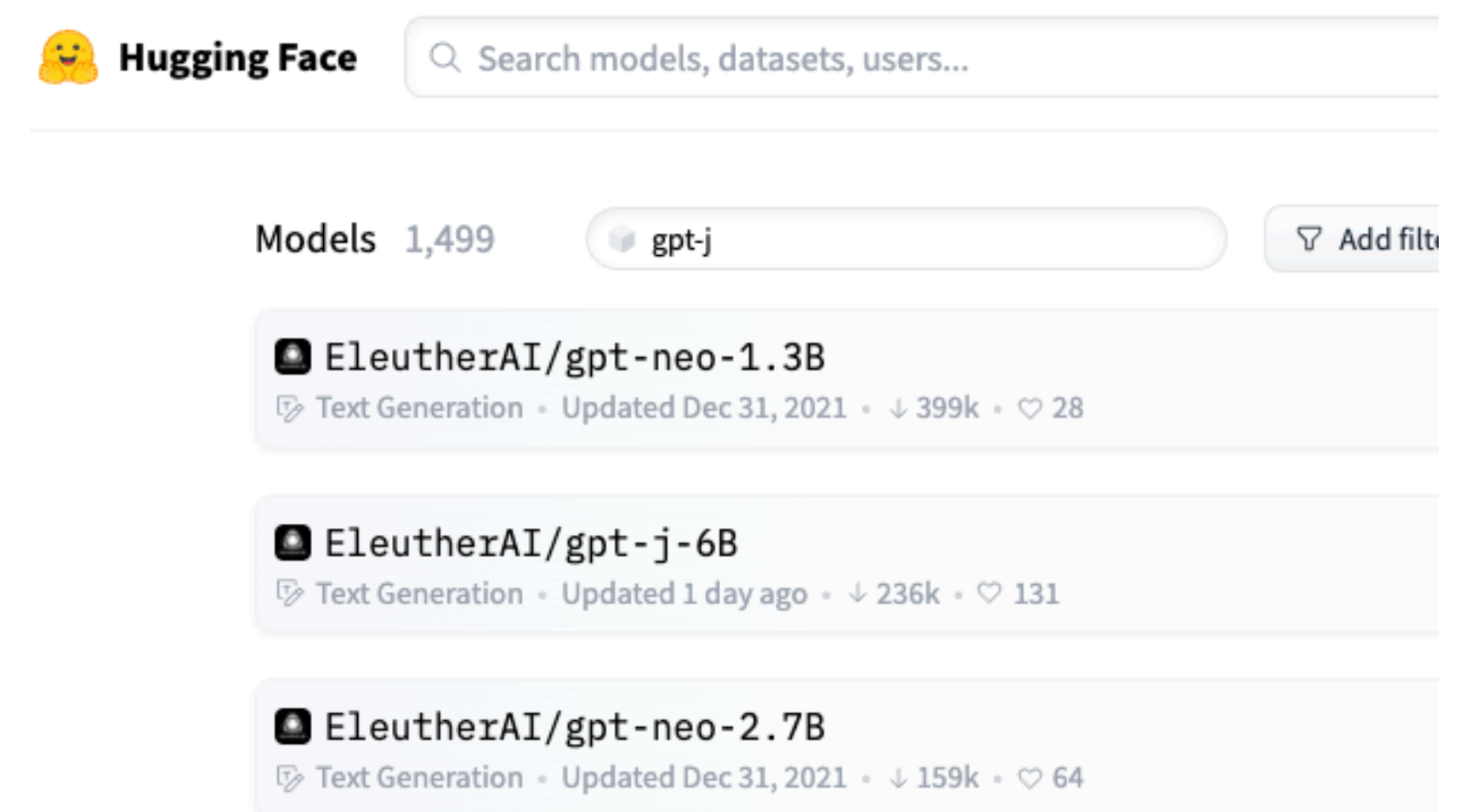
- For a task specific model: how do we even **specify** the control words?

Output

Jupiter is the fifth planet from the Sun. It is a gas giant that is the largest in the Solar System. It is the third brightest object in the night sky. People have been observing it since prehistoric times.

Controlling neural text generation

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

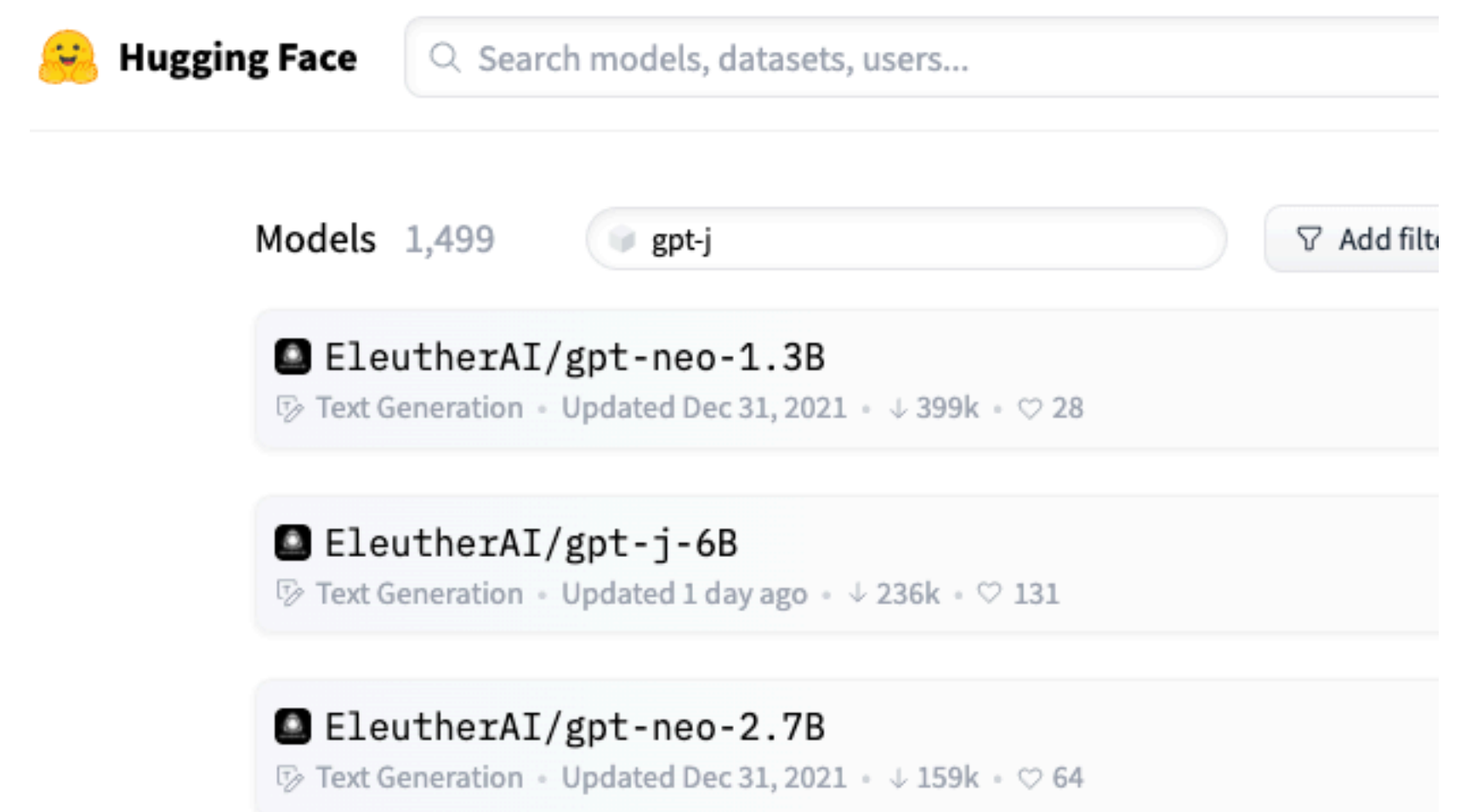


The screenshot shows the Hugging Face model hub interface. At the top, there is a search bar with the text "Search models, datasets, users...". Below the search bar, the text "Models 1,499" is displayed. A filter dropdown menu is open, showing "gpt-j" selected. To the right of the filter, there is a button labeled "Add filters". Below the filter, three model cards are visible, each representing a text generation model by EleutherAI:

- EleutherAI/gpt-neo-1.3B**
Text Generation • Updated Dec 31, 2021 • ↓ 399k • ♥ 28
- EleutherAI/gpt-j-6B**
Text Generation • Updated 1 day ago • ↓ 236k • ♥ 131
- EleutherAI/gpt-neo-2.7B**
Text Generation • Updated Dec 31, 2021 • ↓ 159k • ♥ 64

Controlling neural text generation

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

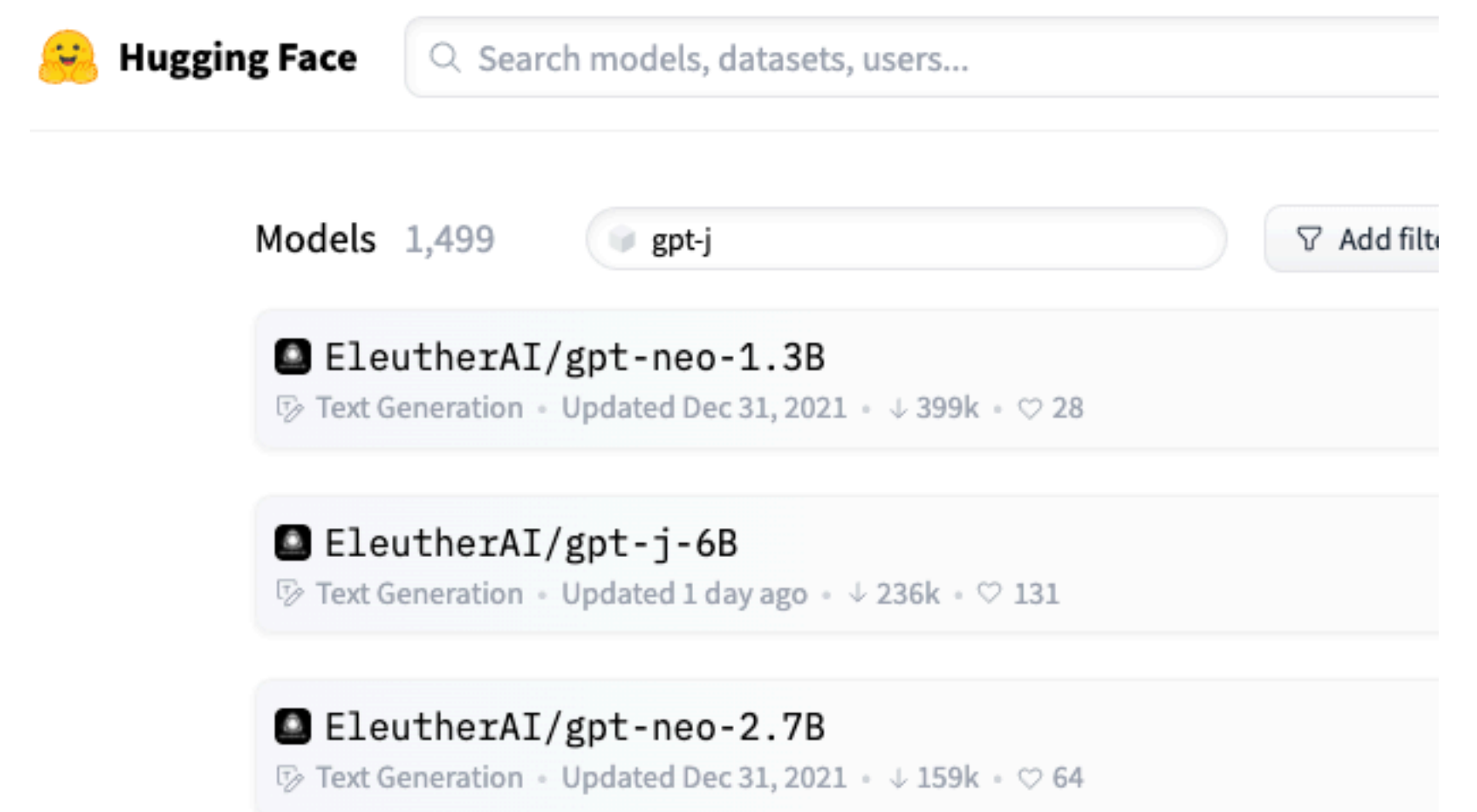


The screenshot shows the Hugging Face model hub interface. At the top, there is a search bar with the text "Search models, datasets, users...". Below the search bar, the text "Models 1,499" is displayed. A filter dropdown menu is open, showing "gpt-j" selected. To the right of the filter, there is a button labeled "Add filters". Below the filter, three model cards are visible, each representing a text generation model by EleutherAI:

- EleutherAI/gpt-neo-1.3B**
Text Generation • Updated Dec 31, 2021 • ↓ 399k • ♥ 28
- EleutherAI/gpt-j-6B**
Text Generation • Updated 1 day ago • ↓ 236k • ♥ 131
- EleutherAI/gpt-neo-2.7B**
Text Generation • Updated Dec 31, 2021 • ↓ 159k • ♥ 64

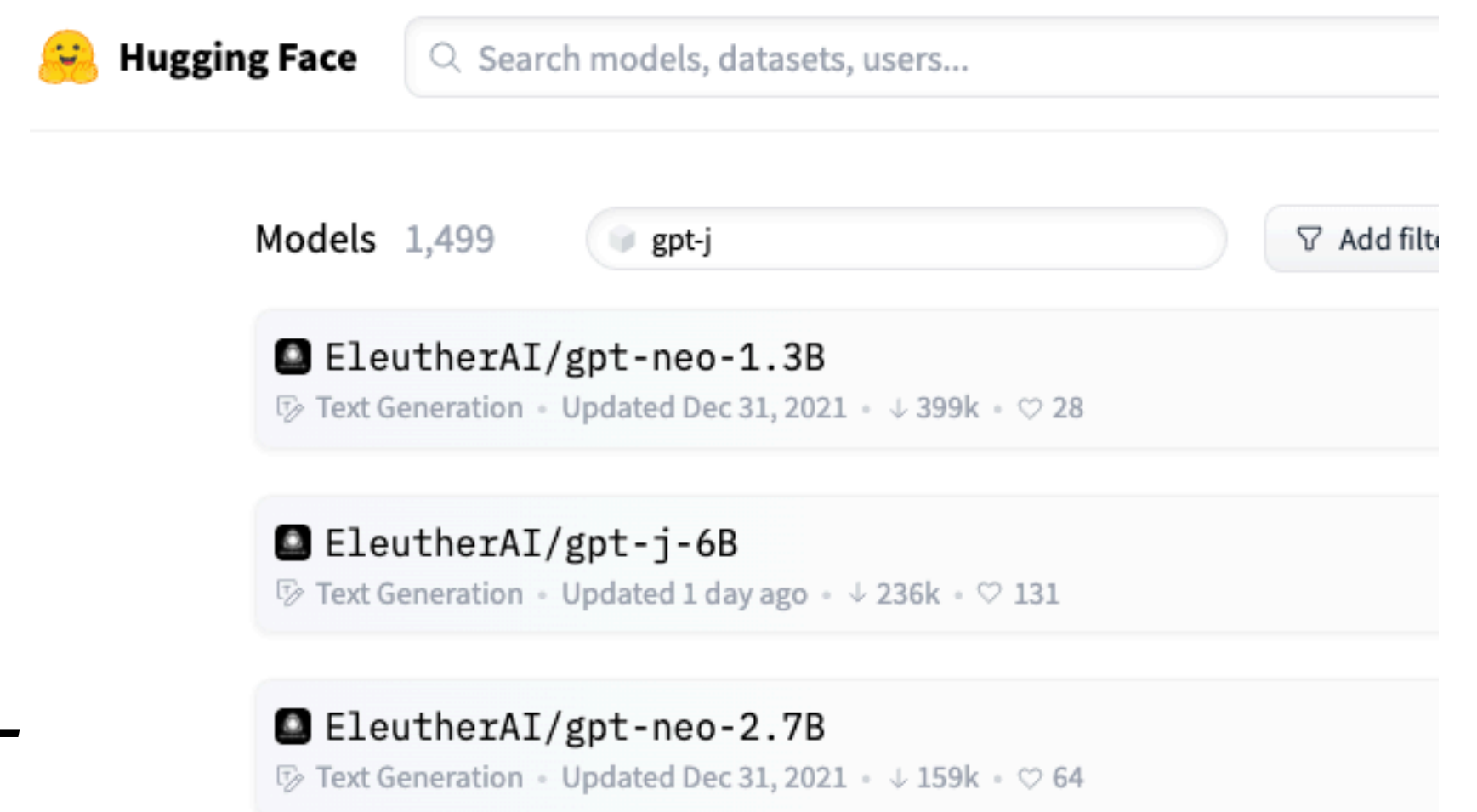
Controlling neural text generation

- 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



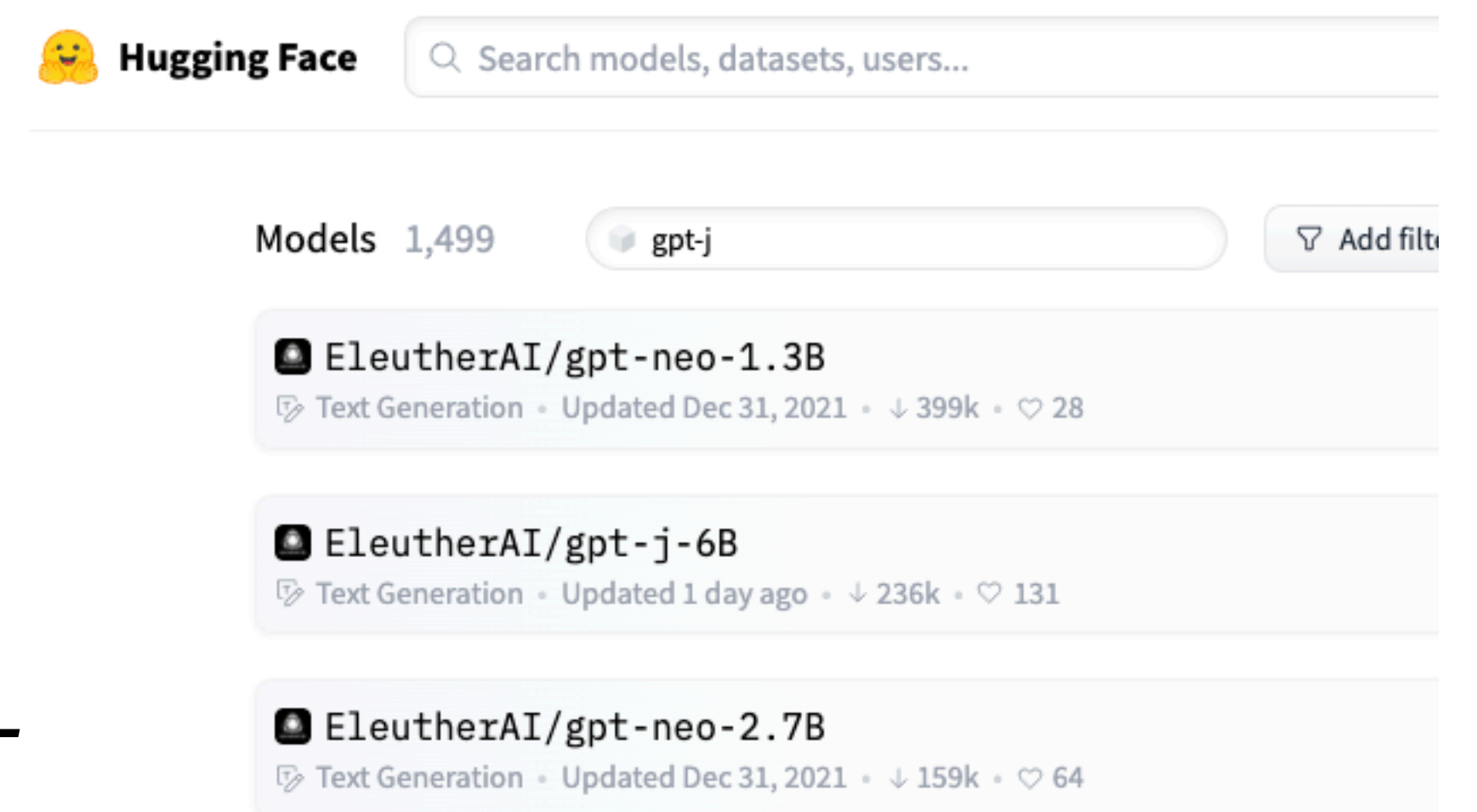
Controlling neural text generation

- 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...)

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

Language Model

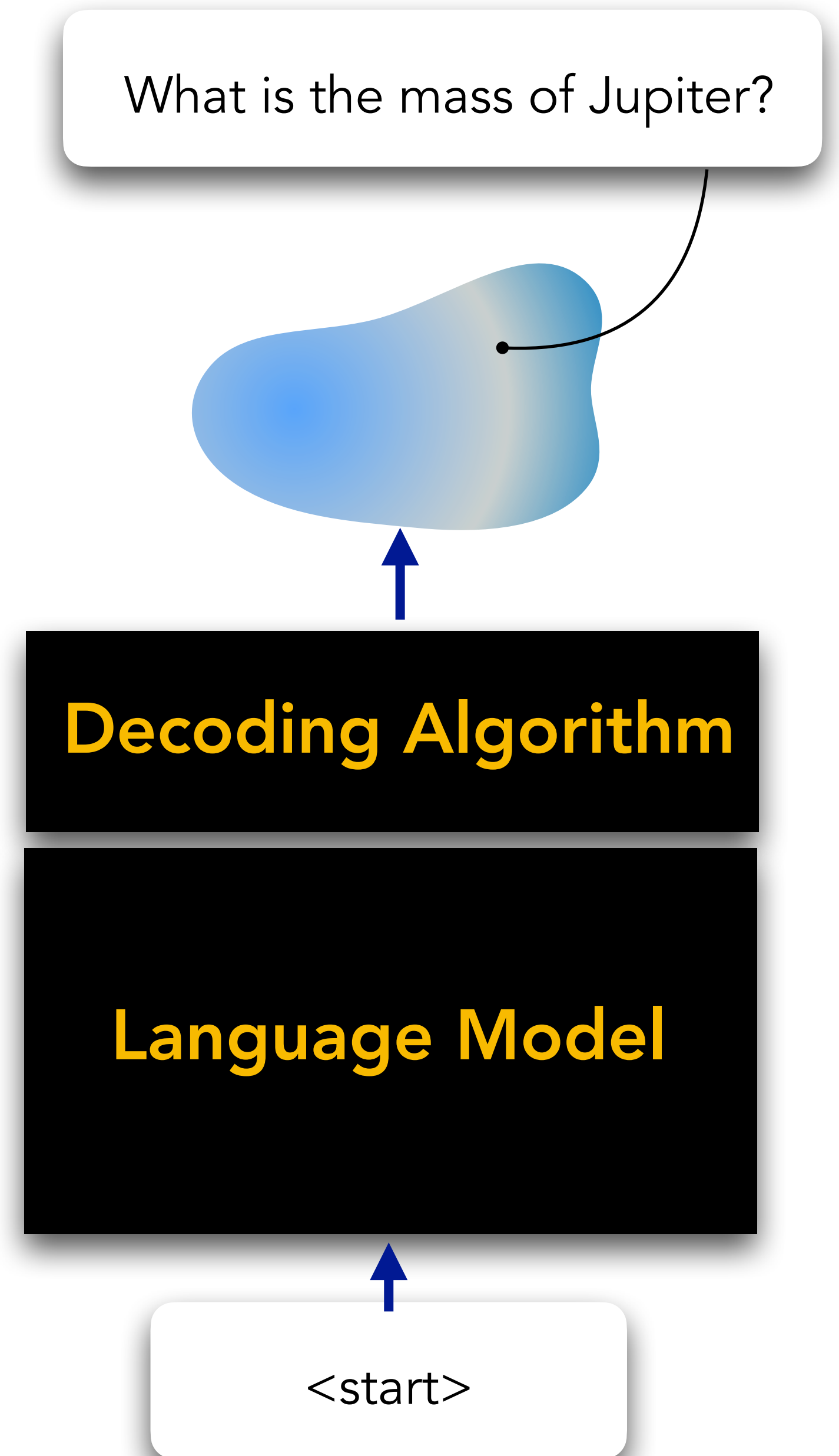
Control through inference

Model + decoding

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

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

- Use an **inference/decoding** algorithm to generate text
 - $\hat{\mathbf{y}} = \text{decode}(p_{\theta}(\cdot | \mathbf{x}))$



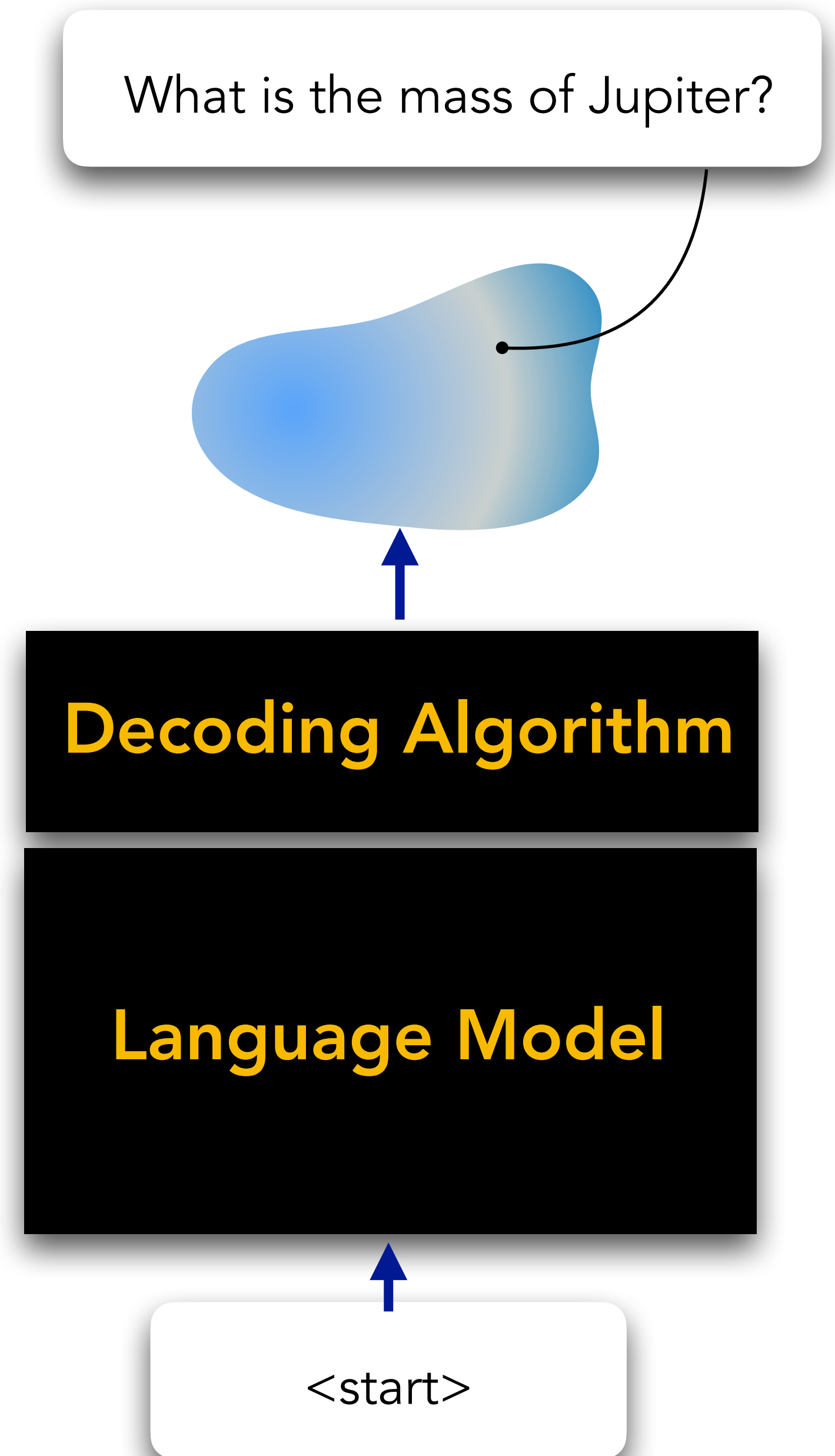
Control through inference

Model + decoding

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

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

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



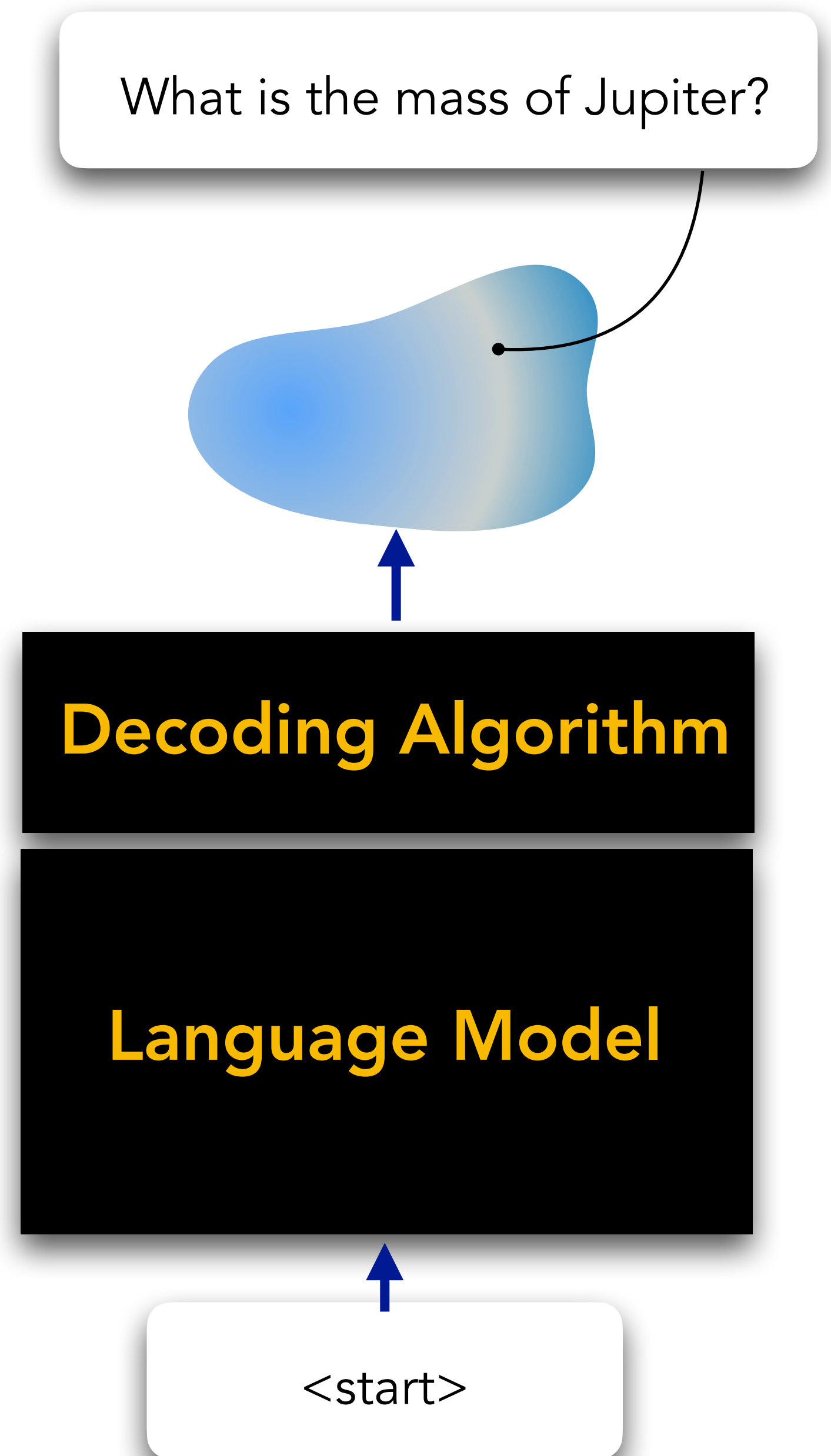
Control through inference

Model + decoding

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

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

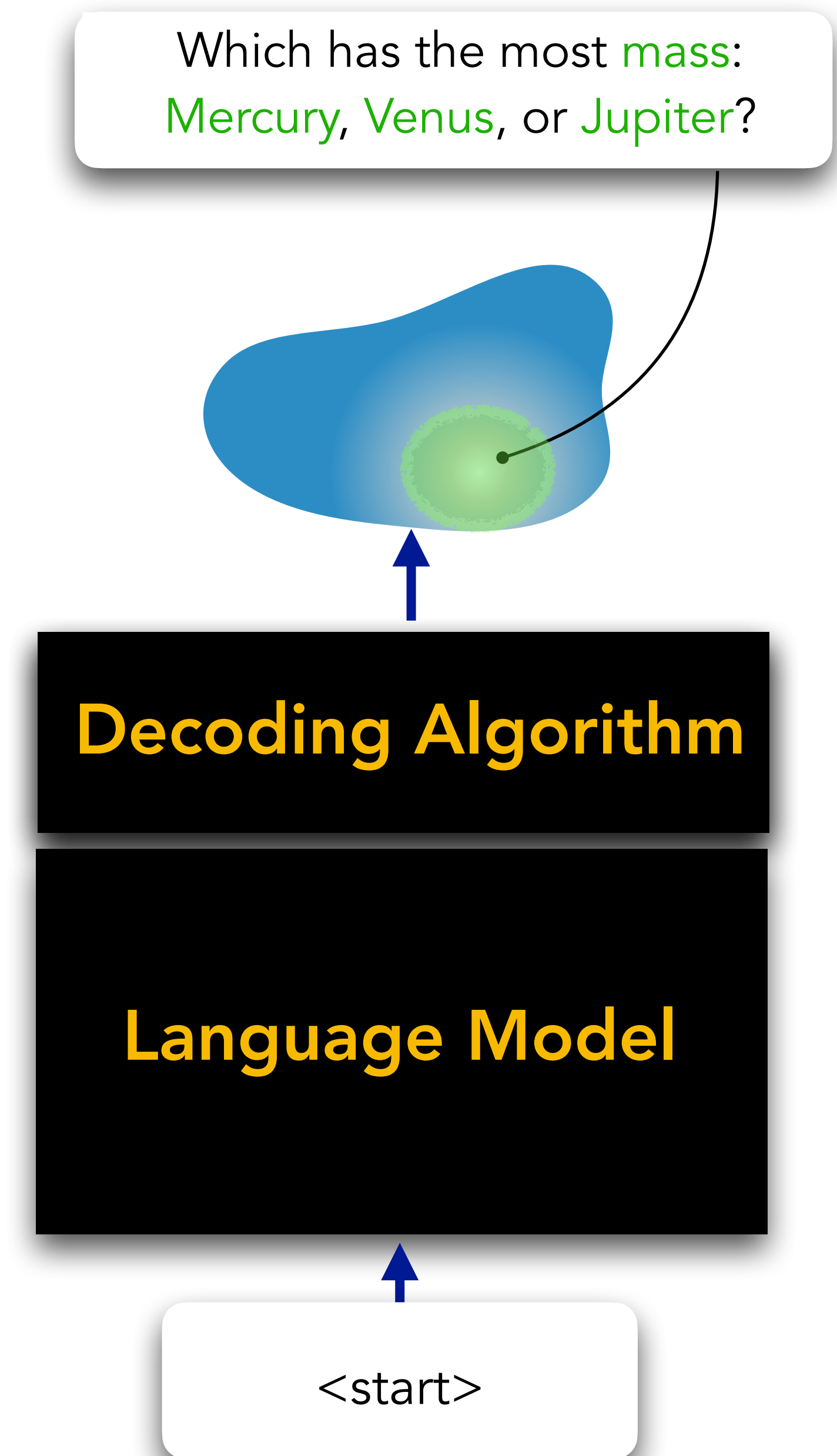
- Use an **inference/decoding** algorithm to generate text
 - $\hat{\mathbf{y}} = \text{decode}(p_{\theta}(\cdot | \mathbf{x}))$
 - e.g. sampling, $\mathbf{y}_t \sim p_{\theta}(y_t | y_{<t}, \mathbf{x})$
 - e.g. maximization $y_t = \arg \max_{y_t} p_{\theta}(y_t | y_{<t}, \mathbf{x})$



Constraints through inference

Model + decoding

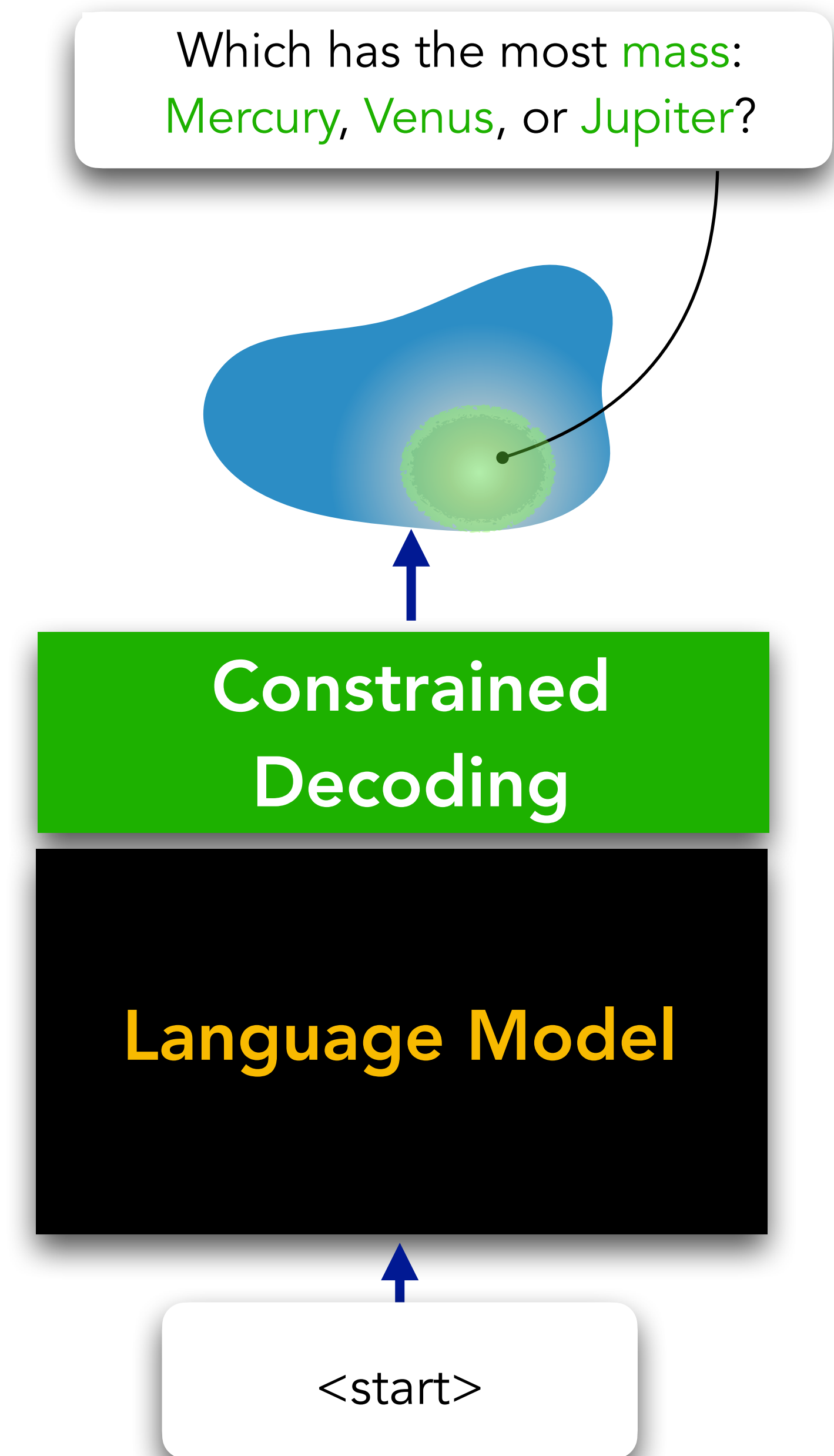
- Control: **constraints** on the generation distribution



Constraints through inference

Model + decoding

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

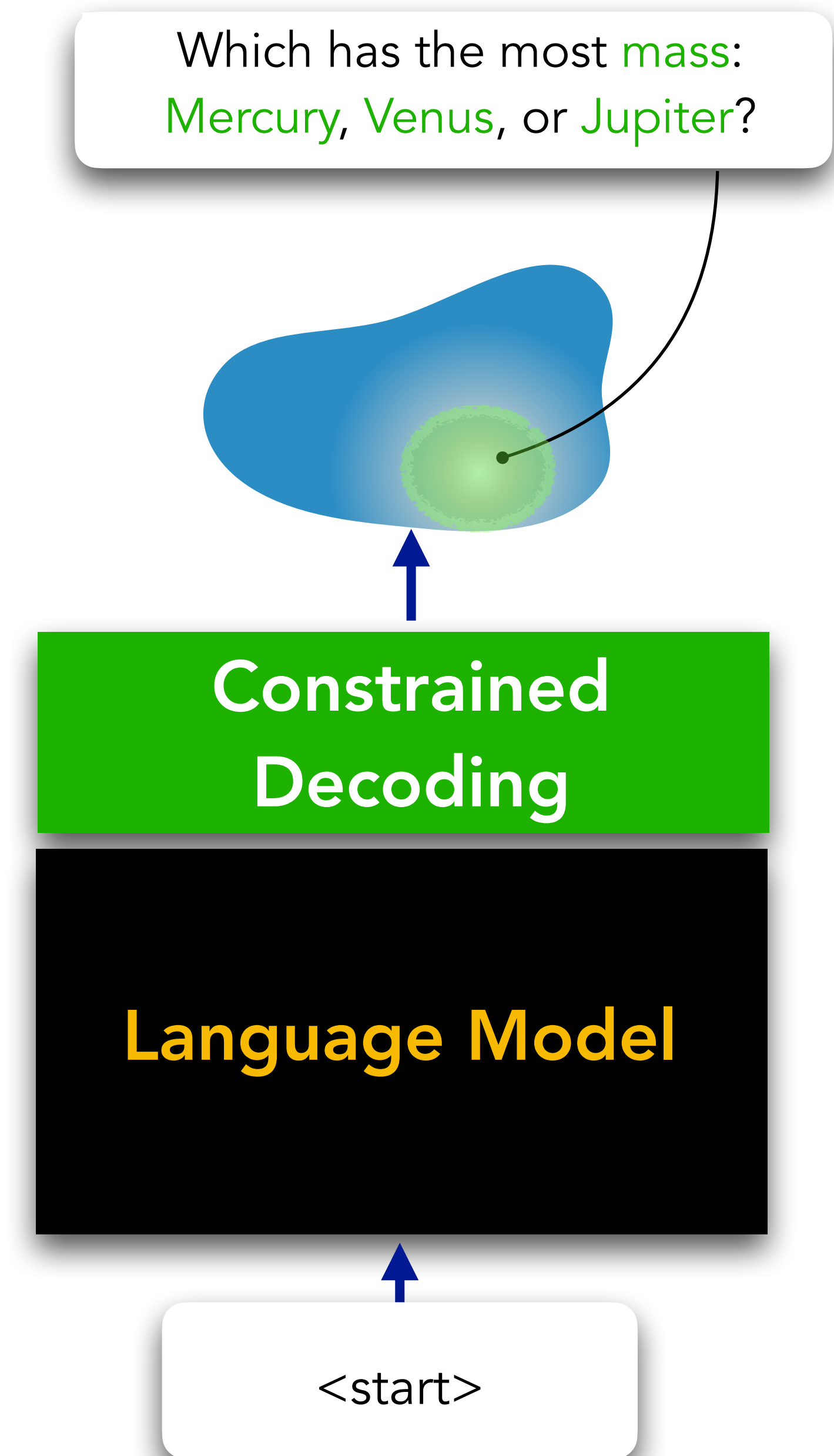


Constraints through inference

Model + decoding

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

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

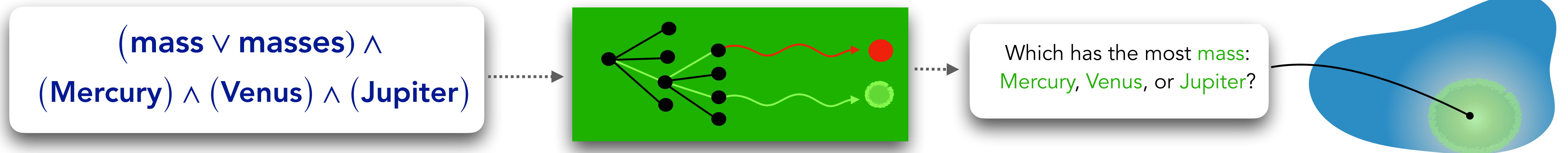


Constrained generation through inference

- Today: decoding algorithms for constrained generation from two perspectives

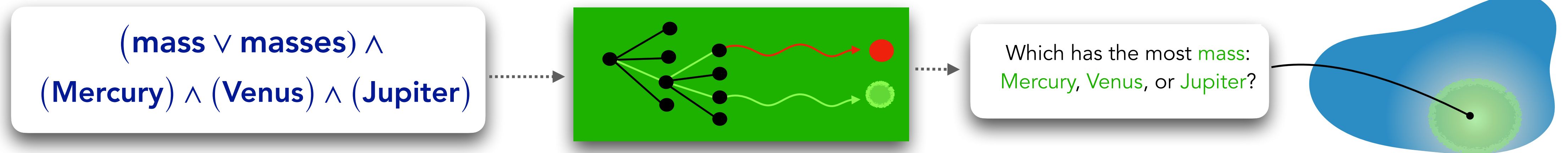
Constrained generation through inference

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

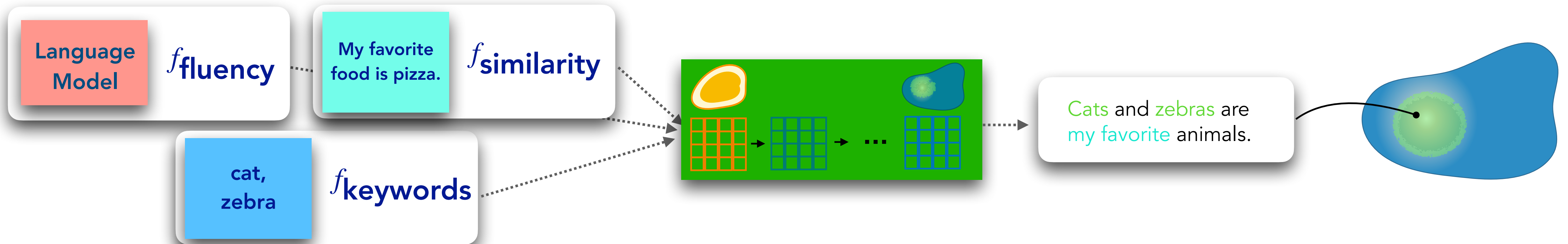


Constrained generation through inference

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



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

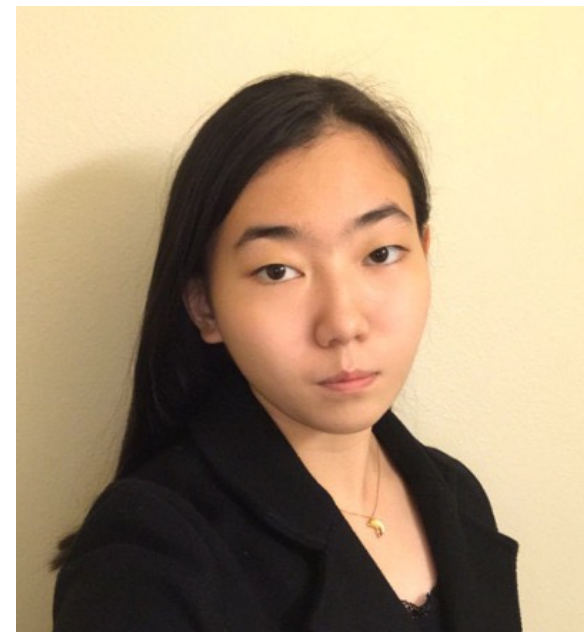


Constrained generation through *discrete* inference

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

In Submission, [arxiv:2112.08726](https://arxiv.org/abs/2112.08726)

Ximing Lu



Sean Welleck



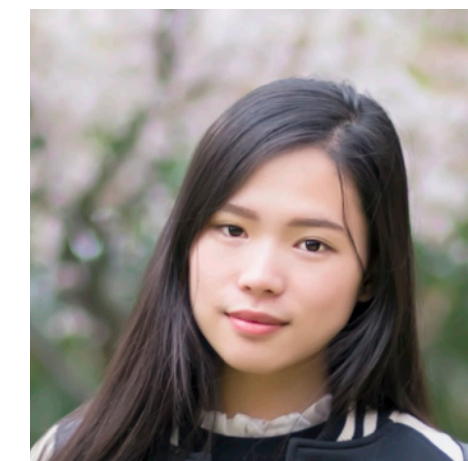
Peter West



Liwei Jiang



Lianhui Qin



Youngjae Yu



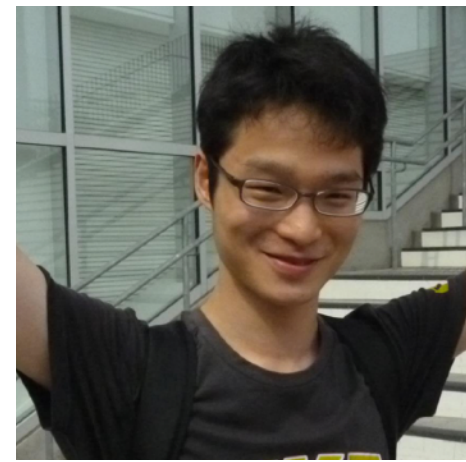
Yejin Choi



Daniel Khashabi



Jungo Kasai



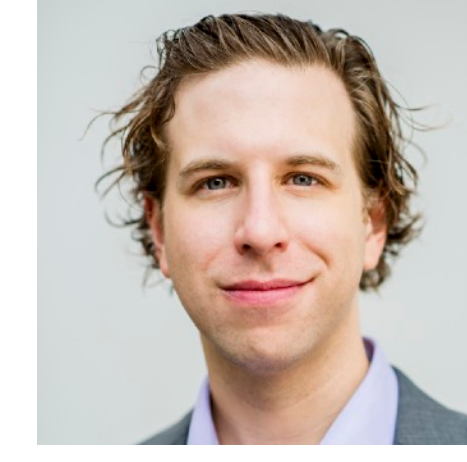
Ronan Le Bras



Rowan Zellers



Noah Smith



Logical lexical constraints

- Ensure certain words **appear** or **do not appear**

Generate a sentence using
cat and **fish**, but **not dog**

Logical Constraints

$(\text{cat} \vee \text{cats}) \wedge (\text{fish}) \wedge (\neg \text{dog})$

C

The **cat** jumped on the table and saw a **fish**.

A*-NeuroLogic

**Off-the-shelf
Language Model**

<start>

Decoding Objective

- Goal: $\mathbf{y}_* = \arg \max_{\mathbf{y} \in \mathcal{Y}} \underbrace{\log p_{\theta}(\mathbf{y})}_{\text{fluency}} + \underbrace{C(\mathbf{y})}_{\text{constraints}}$

Logical Constraints
 $(\text{cat} \vee \text{cats}) \wedge (\text{fish}) \wedge (\neg \text{dog})$

Standard decoding

Beam search

Standard decoding

Beam search

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

Standard decoding

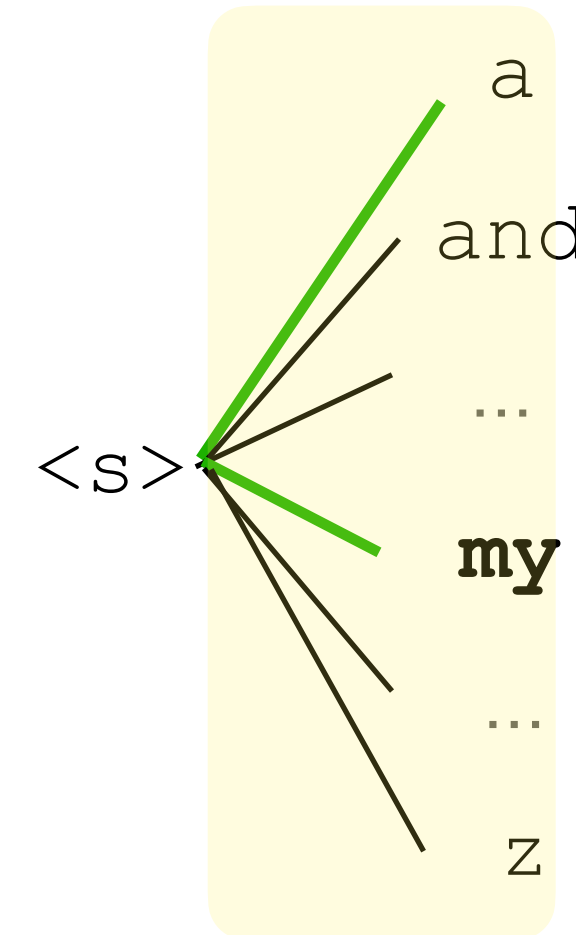
Beam search

- $y_* \approx \arg \max_{y \in \mathcal{Y}} \underbrace{\log p_\theta(y)}_{\text{fluency}} + \underbrace{0}_{\text{constraints}} \quad \langle s \rangle$
- Left-to-right search on the lattice of tokens:

Standard decoding

Beam search

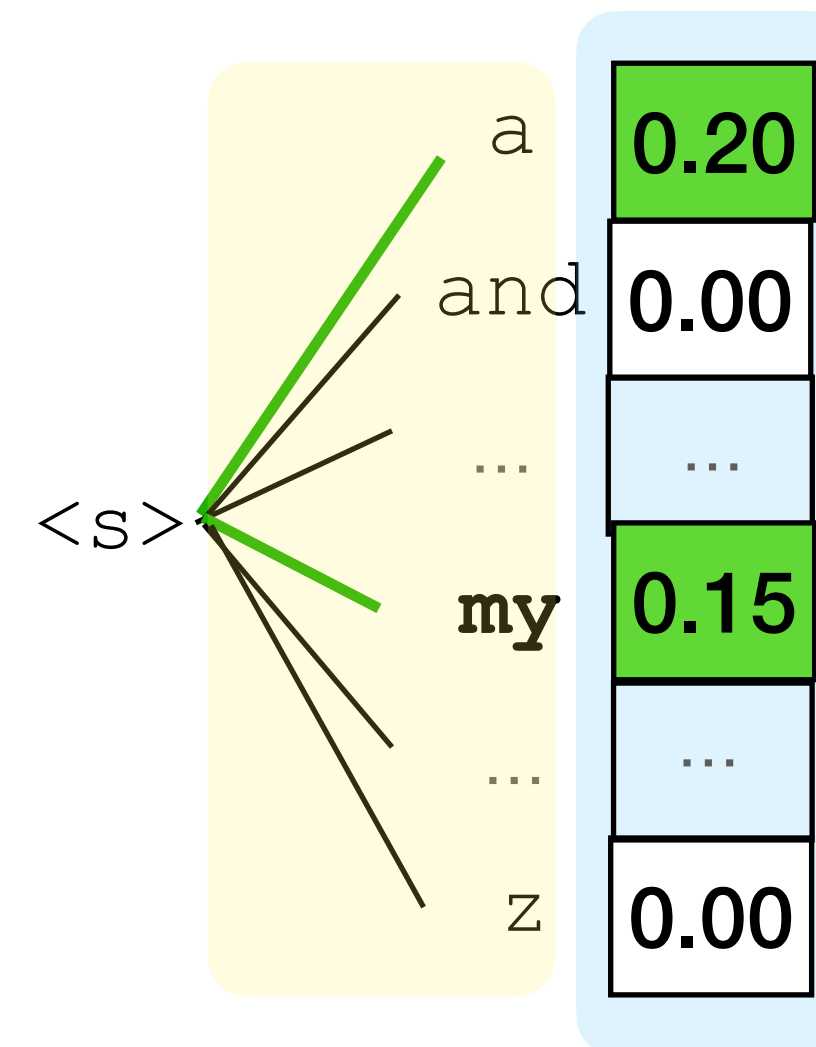
- $y_* \approx \arg \max_{y \in \mathcal{Y}} \underbrace{\log p_\theta(y)}_{\text{fluency}} + \underbrace{0}_{\text{constraints}}$
- Left-to-right search on the lattice of tokens:
 - Expand prefixes with next-tokens



Standard decoding

Beam search

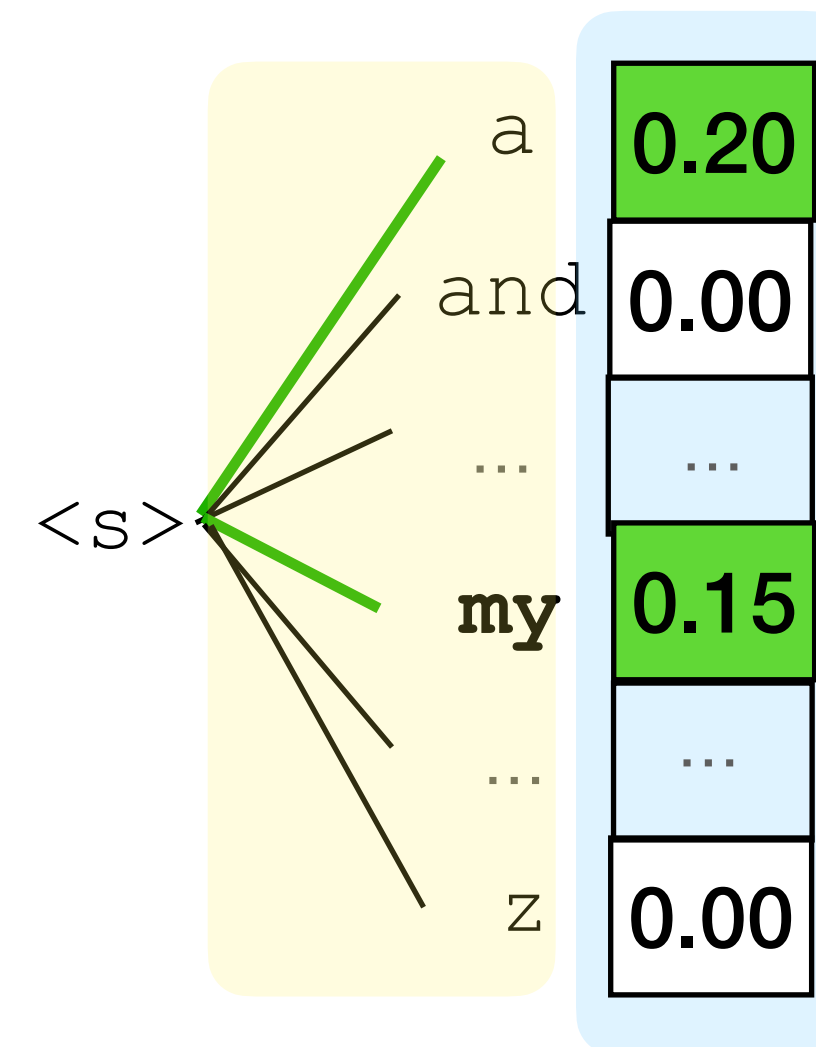
- $y_* \approx \arg \max_{y \in \mathcal{Y}} \underbrace{\log p_\theta(y)}_{\text{fluency}} + \underbrace{0}_{\text{constraints}}$
- Left-to-right search on the lattice of tokens:
 - Expand prefixes with next-tokens
 - Score each using $\underbrace{\log p_\theta(y_t | y_{<t})}_{\text{fluency}}$



Standard decoding

Beam search

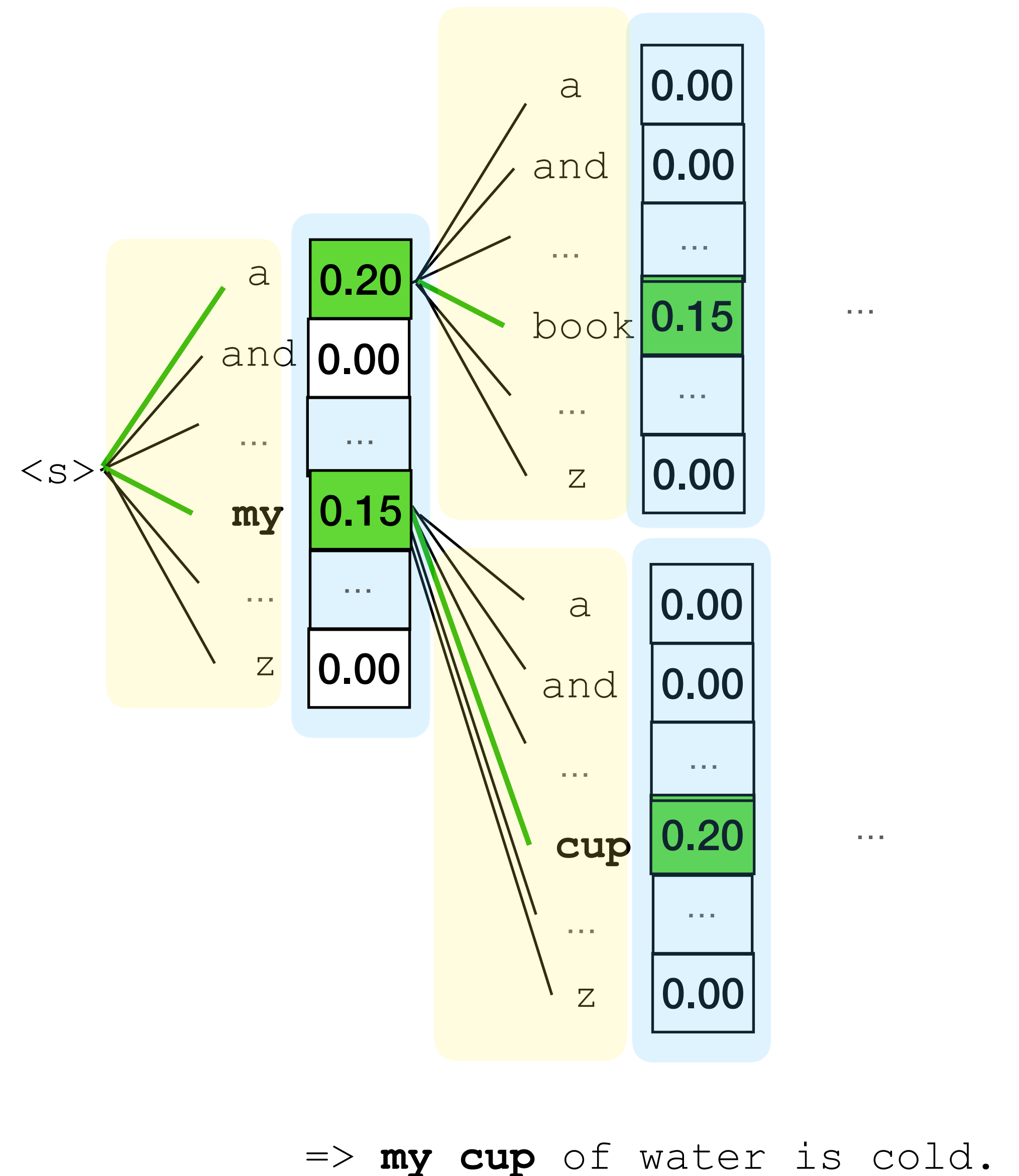
- $y_* \approx \arg \max_{y \in \mathcal{Y}} \underbrace{\log p_\theta(y)}_{\text{fluency}} + \underbrace{0}_{\text{constraints}}$
- Left-to-right search on the lattice of tokens:
 - Expand prefixes with next-tokens
 - Score each using $\underbrace{\log p_\theta(y_t | y_{<t})}_{\text{fluency}}$
 - Select the k best, and repeat



Standard decoding

Beam search

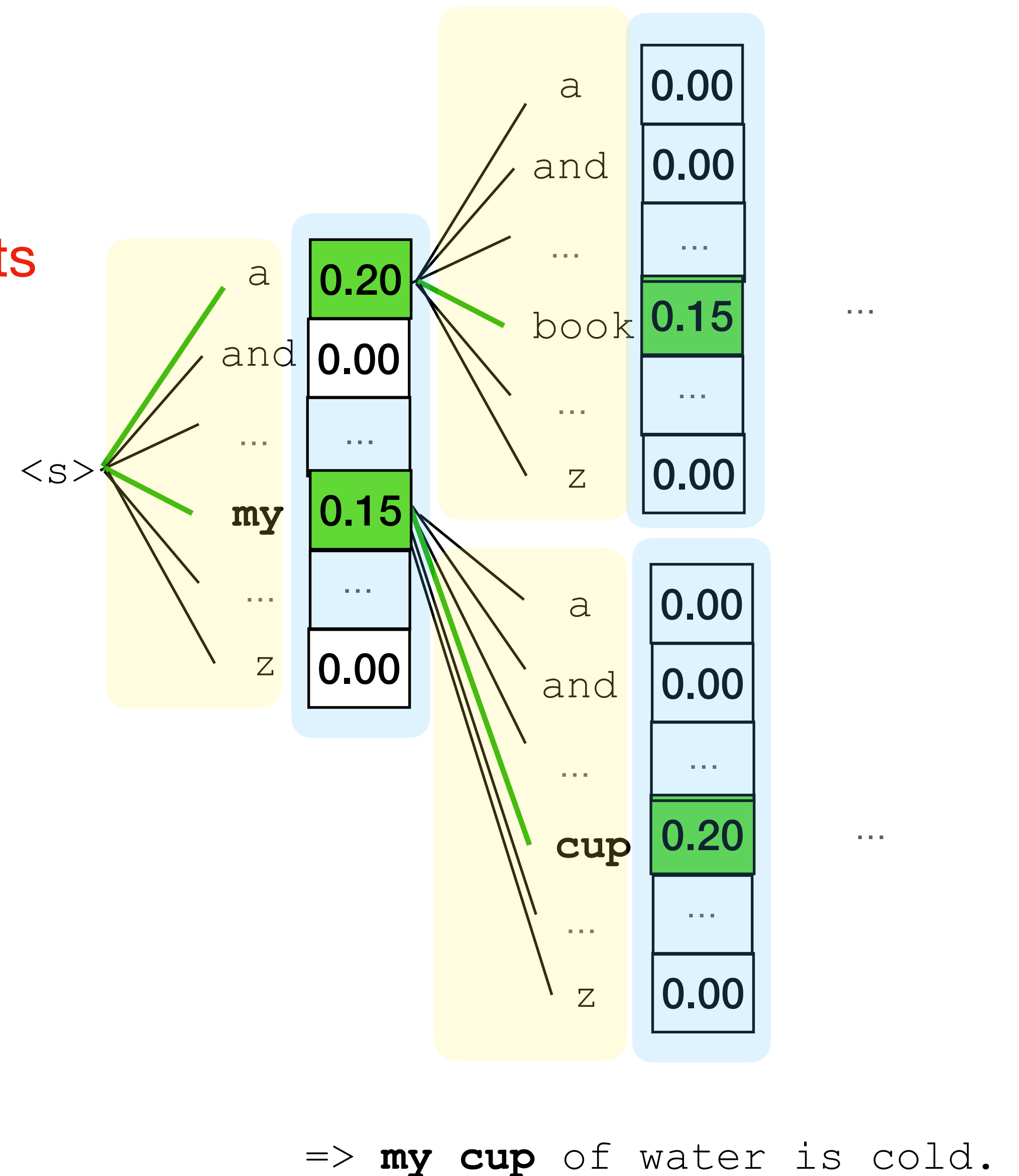
- $y_* \approx \arg \max_{y \in \mathcal{Y}} \underbrace{\log p_\theta(y)}_{\text{fluency}} + \underbrace{0}_{\text{constraints}}$
- Left-to-right search on the lattice of tokens:
 - Expand prefixes with next-tokens
 - Score each using $\underbrace{\log p_\theta(y_t | y_{<t})}_{\text{fluency}}$
 - Select the k best, and repeat



Standard decoding

Beam search

- $y_* \approx \arg \max_{y \in \mathcal{Y}} \underbrace{\log p_\theta(y)}_{\text{fluency}} + \underbrace{0}_{\text{constraints}}$ Ignores constraints
- Left-to-right search on the lattice of tokens:
 - Expand prefixes with next-tokens
 - Score each using $\underbrace{\log p_\theta(y_t | y_{<t})}_{\text{fluency}}$
 - Select the k best, and repeat

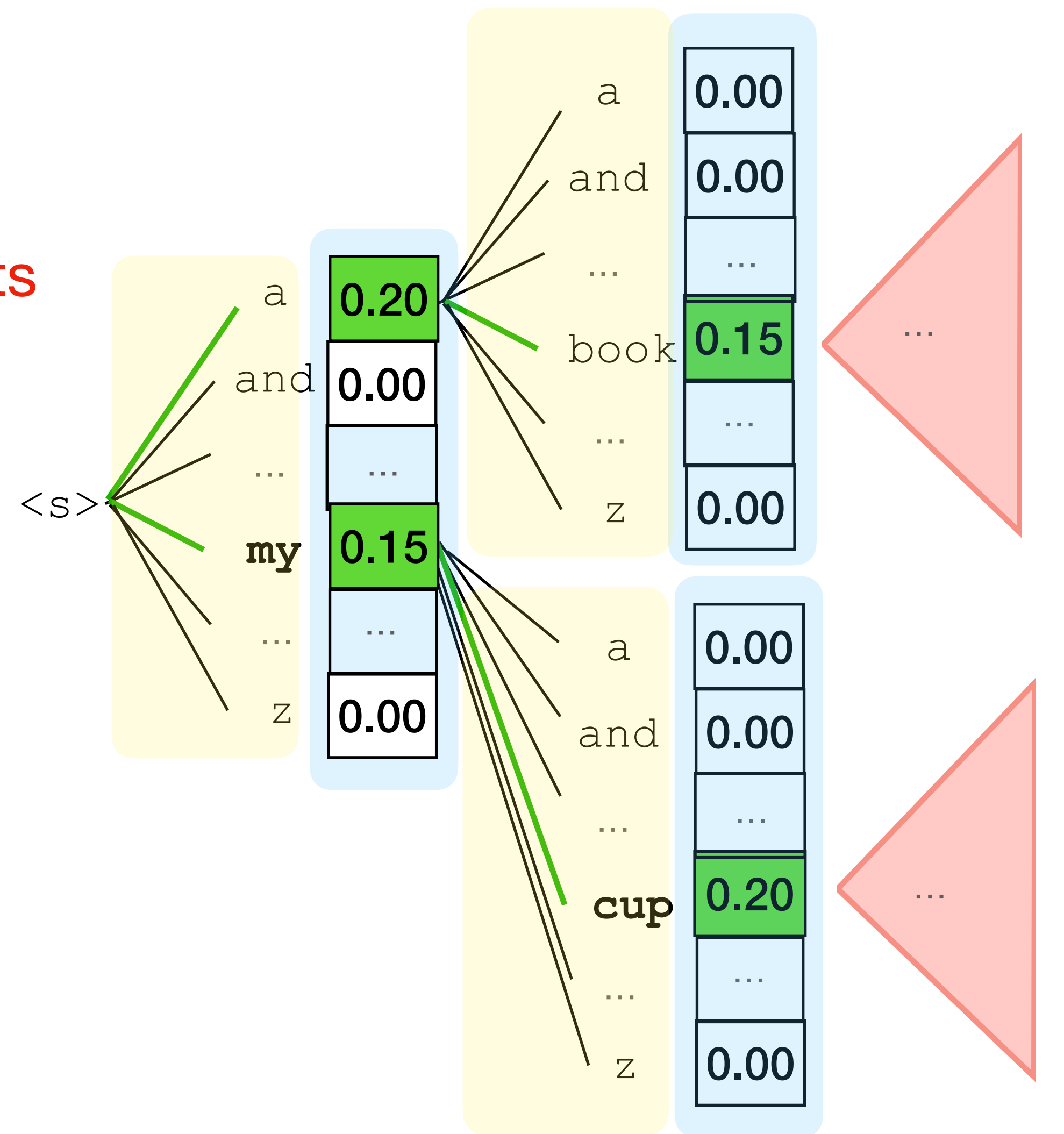


Standard decoding

Beam search

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

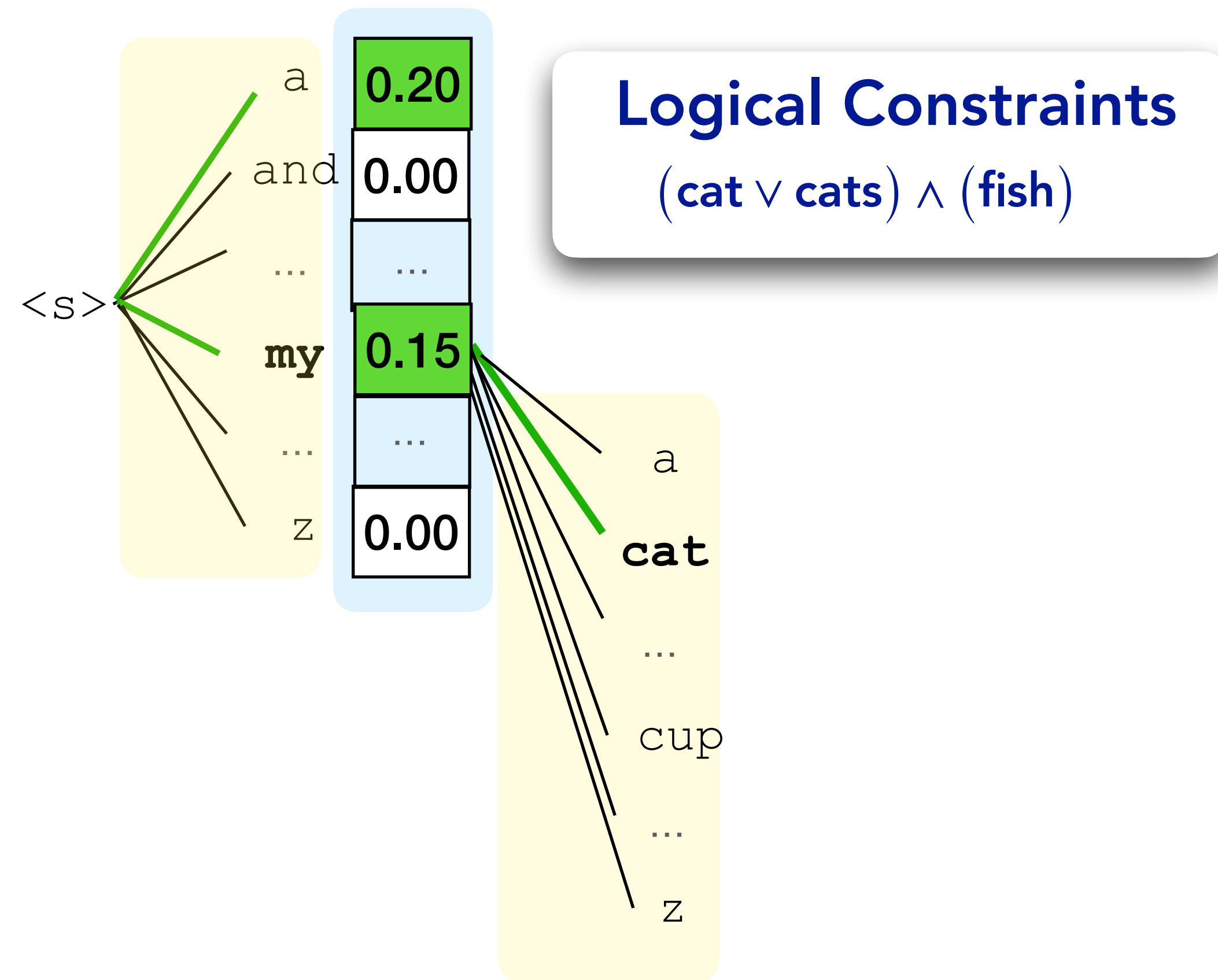
Ignores constraints
- Left-to-right search on the lattice of tokens:
 - Expand prefixes with next-tokens
 - Score each using $\underbrace{\log p_\theta(y_t | y_{<t})}_{\text{fluency}}$ Myopic
 - Select the k best, and repeat



=> **my cup** of water is cold.

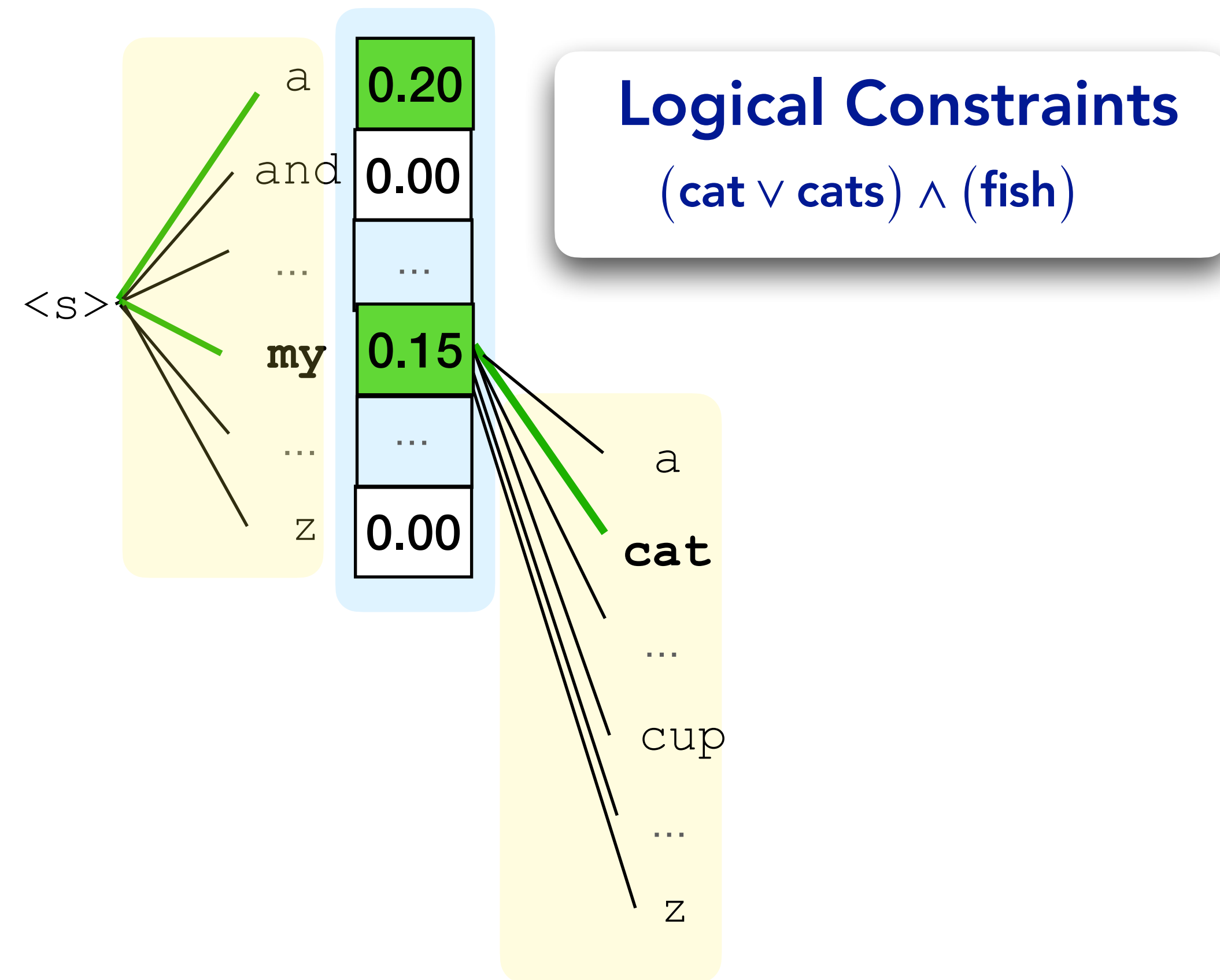
NeuroLogic decoding [Lu et al 2021]

- + favor tokens that [partially] satisfy constraints



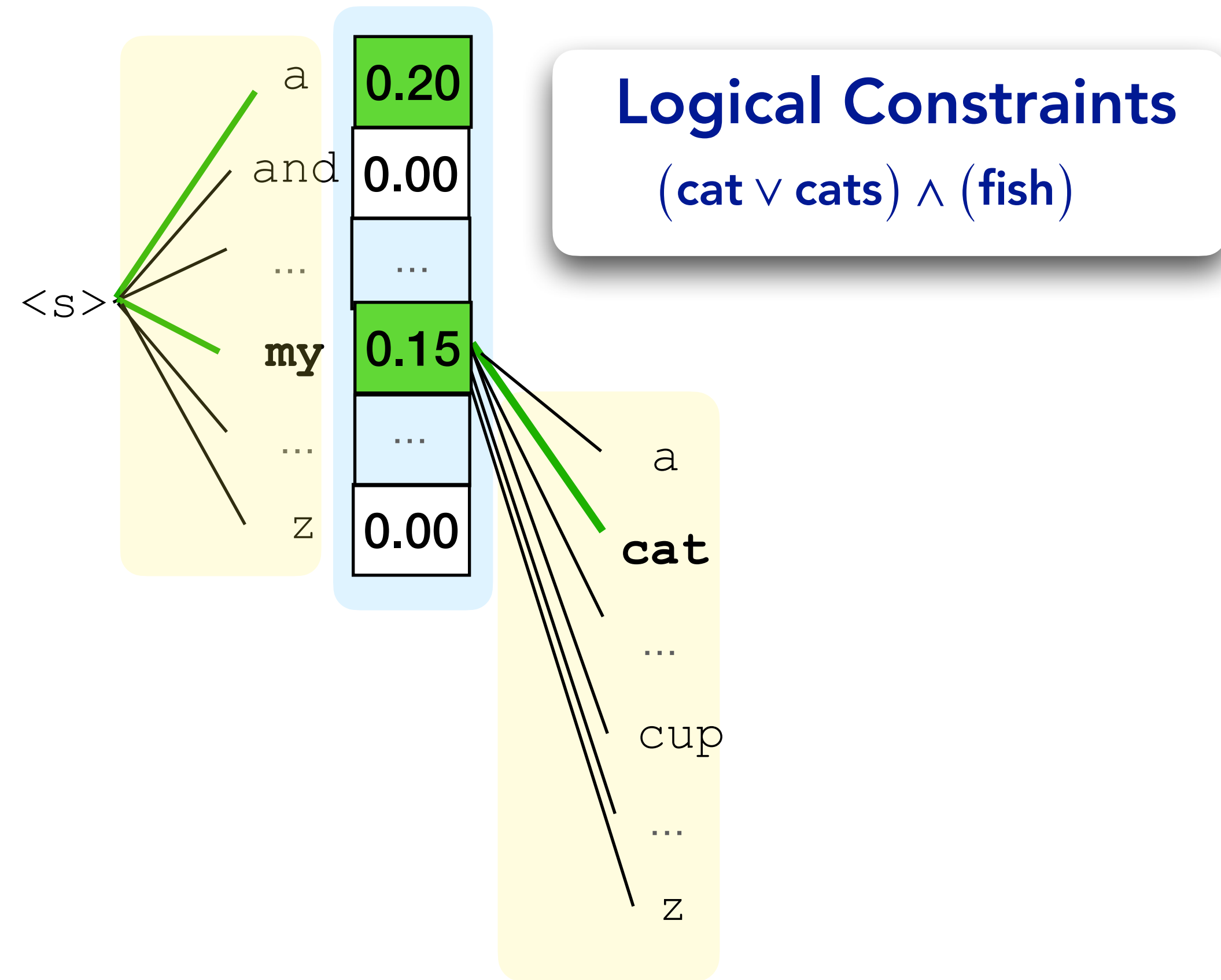
NeuroLogic decoding [Lu et al 2021]

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



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

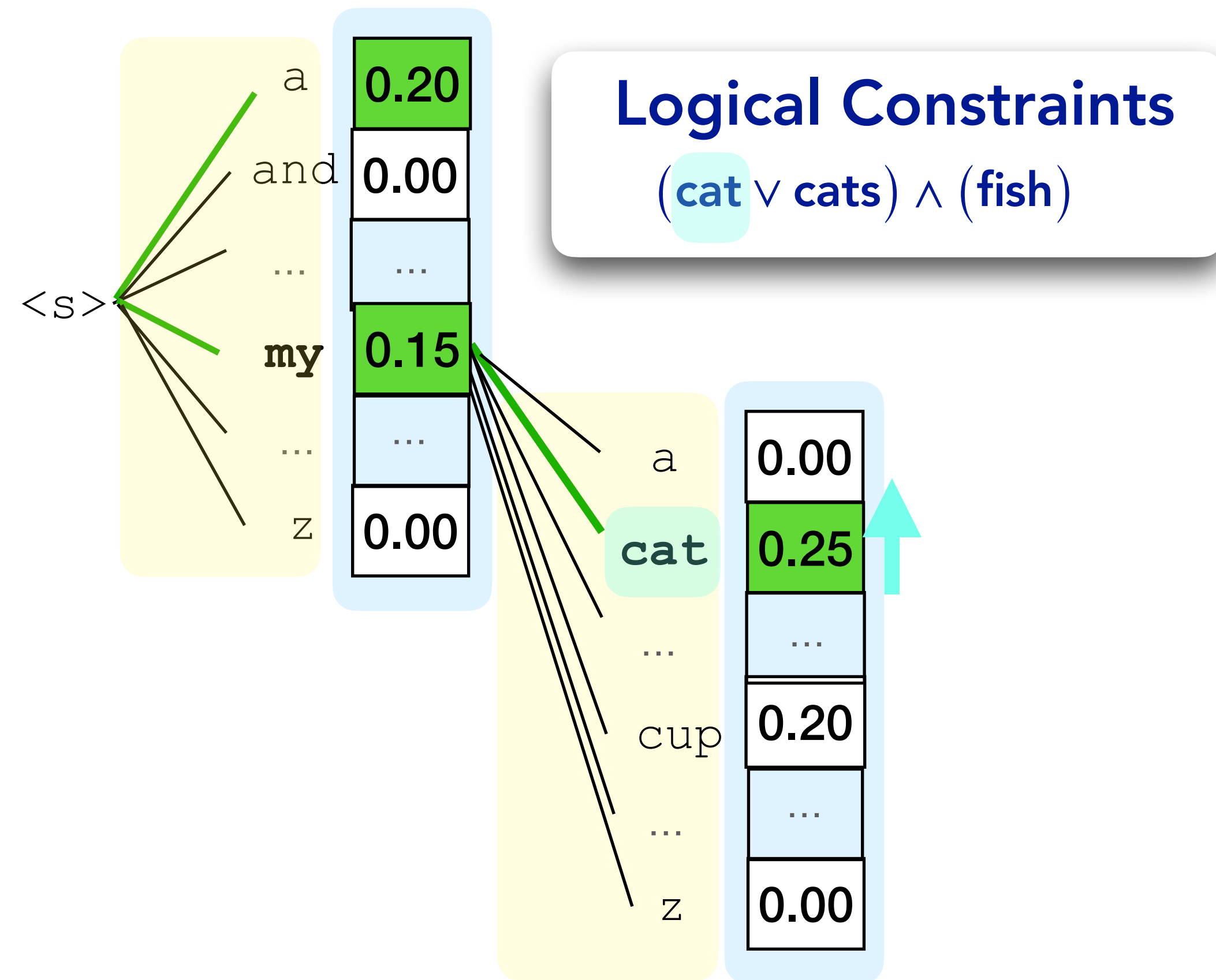
NeuroLogic decoding [Lu et al 2021]

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

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

Not trivial!

Details & other features
out of scope for this talk



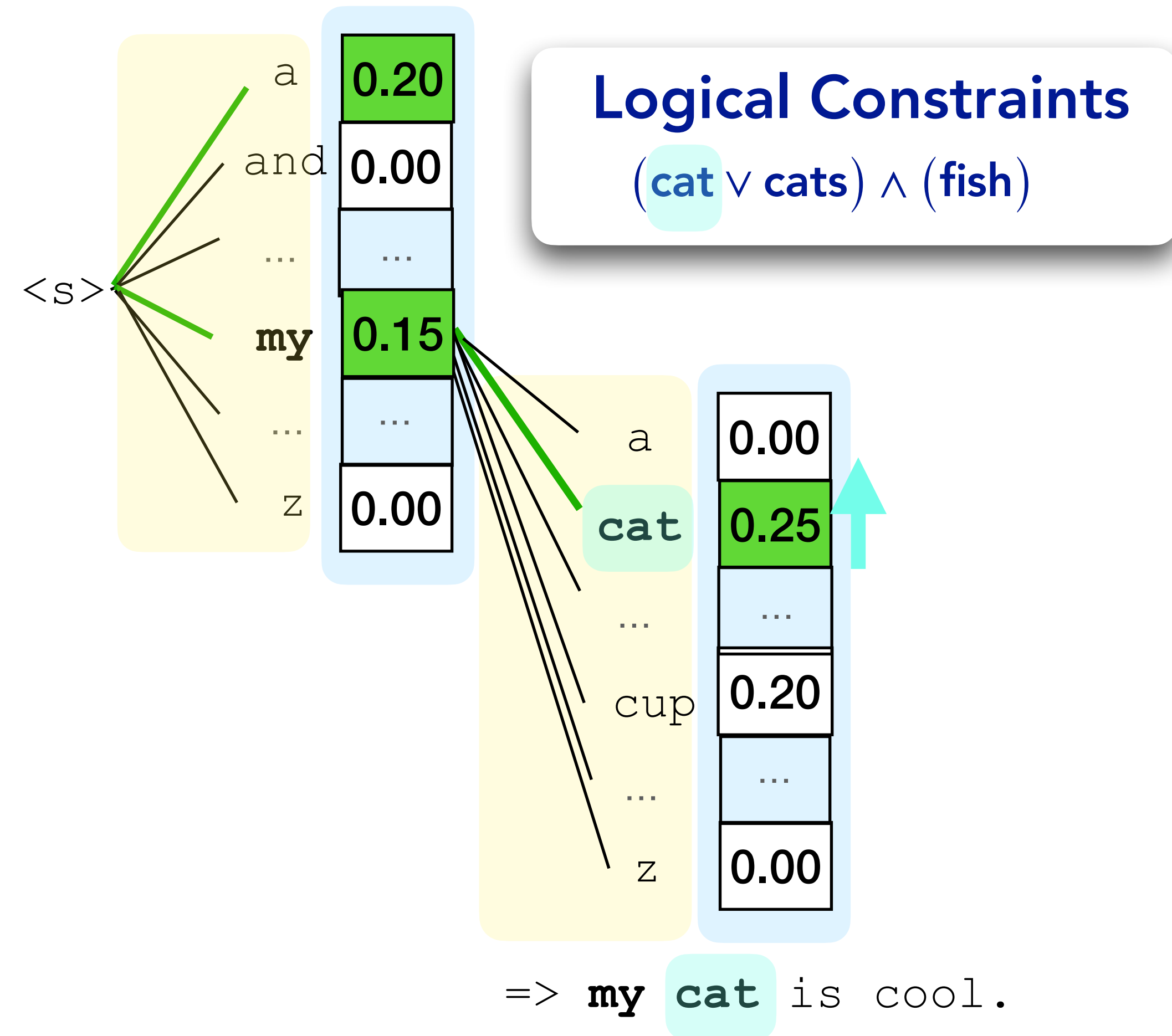
NeuroLogic decoding [Lu et al 2021]

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

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

Not trivial!

Details & other features out of scope for this talk



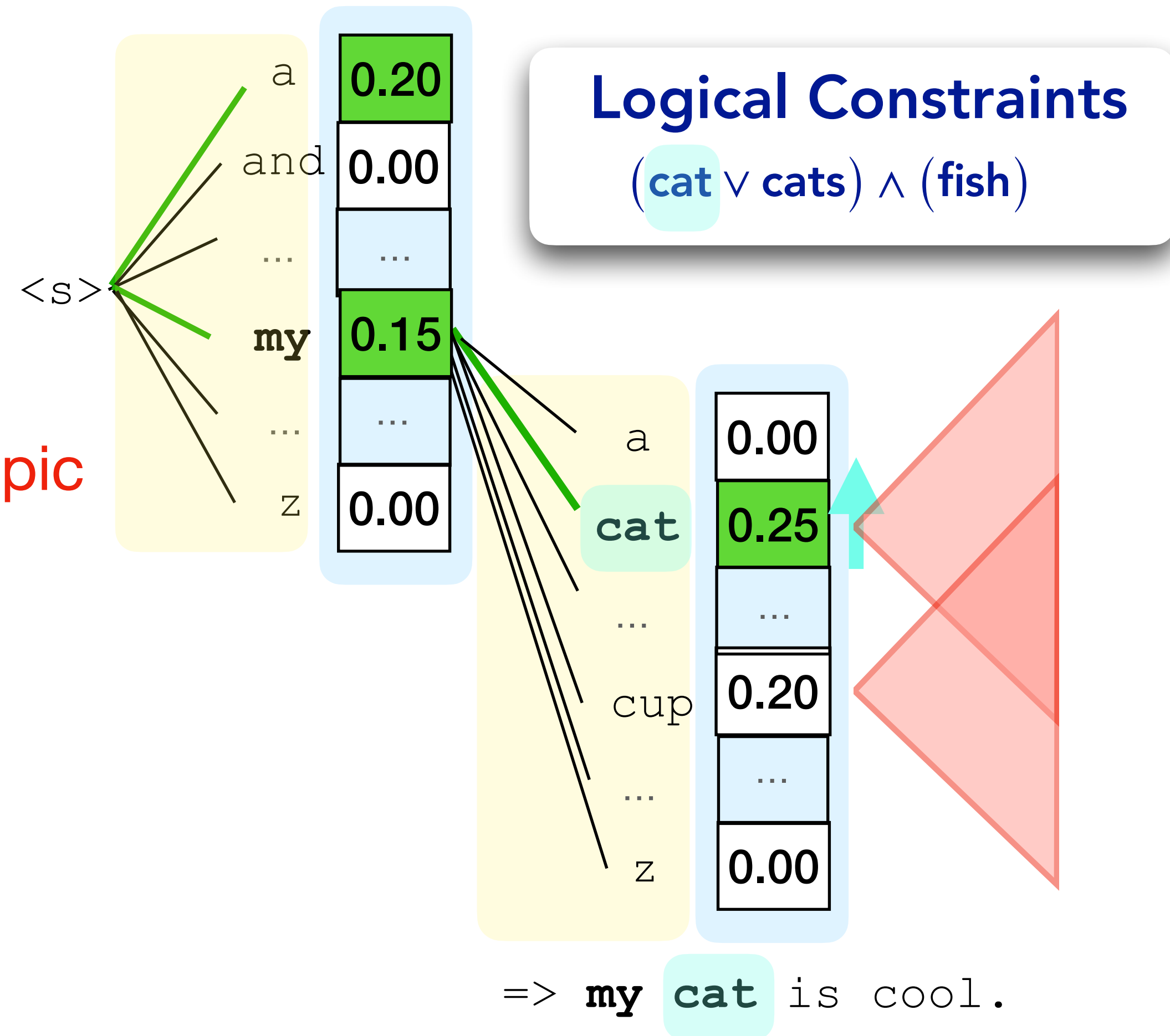
NeuroLogic decoding [Lu et al 2021]

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

$$\underbrace{\log p_{\theta}(y_t | y_{<t})}_{\text{fluency}} + \lambda \underbrace{\max_{c \in \text{remaining}} c(y_t)}_{\text{constraints}} \quad \text{Myopic}$$

Not trivial!

Details & other features out of scope for this talk



NeuroLogic A*esque decoding

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

- $\text{argtopk}_{y_t} \left(\max_{y_{>t}} F(\mathbf{y}_{<t}, y_t, \mathbf{y}_{>t}) \right)$, $F = \text{fluency} + \text{constraints}$

NeuroLogic A*esque decoding

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

$$\bullet \operatorname{argtopk}_{y_t} \left(\max_{y_{>t}} F(y_{<t}, y_t, y_{>t}) \right), F = \text{fluency} + \text{constraints}$$

Intractable

NeuroLogic A*esque decoding

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

- $\text{argtopk}_{y_t} \left(\max_{y_{>t}} F(y_{<t}, y_t, y_{>t}) \right)$, $F = \text{fluency} + \text{constraints}$

Intractable

- A* Search: best-first search with future heuristics

- $f(a) = \underbrace{s(a)}_{\text{score so-far}} + \underbrace{h(a)}_{\text{future heuristic}}$



NeuroLogic A*esque decoding

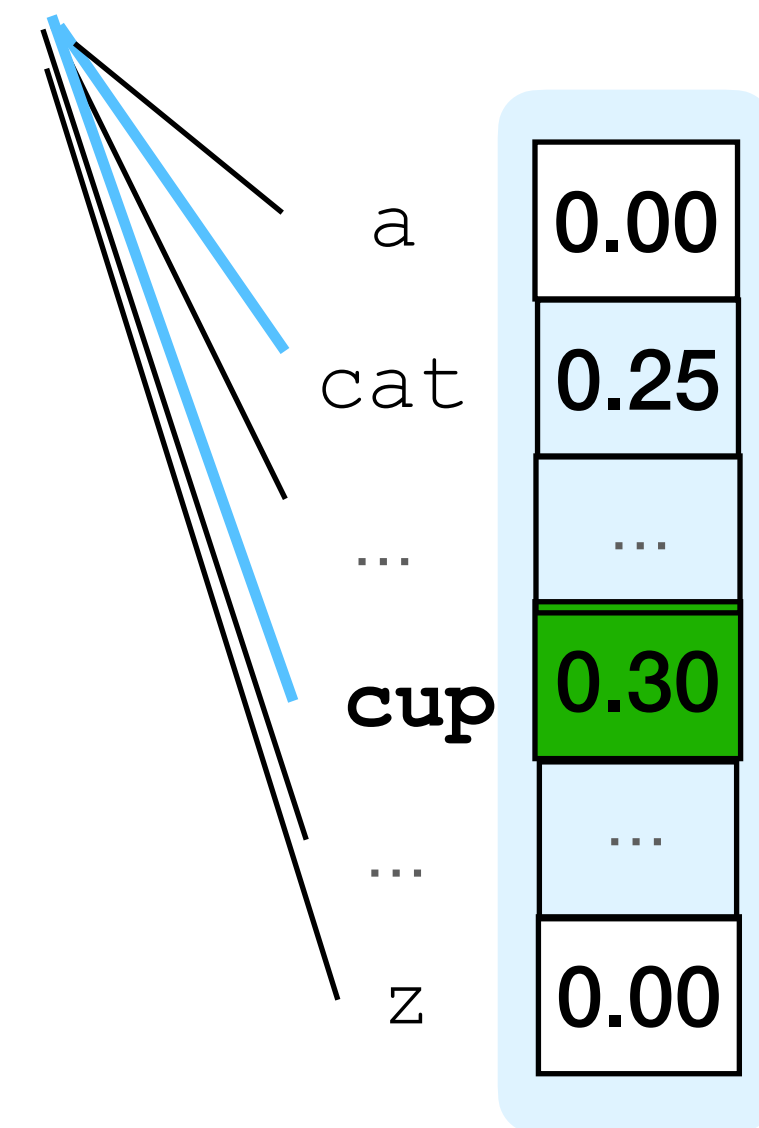
- Approximate with a *lookahead heuristic*:

$$\bullet \operatorname{argtopk}_{y_t} \left(s(\mathbf{y}_{\leq t}) + \right.$$

Fluency + constraints-so-far

Logical Constraints

$(\text{cat} \vee \text{cats}) \wedge (\text{fish})$



NeuroLogic A*esque decoding

- Approximate with a *lookahead heuristic*:

$$\bullet \operatorname{argtopk}_{y_t} \left(s(\mathbf{y}_{\leq t}) + \right.$$

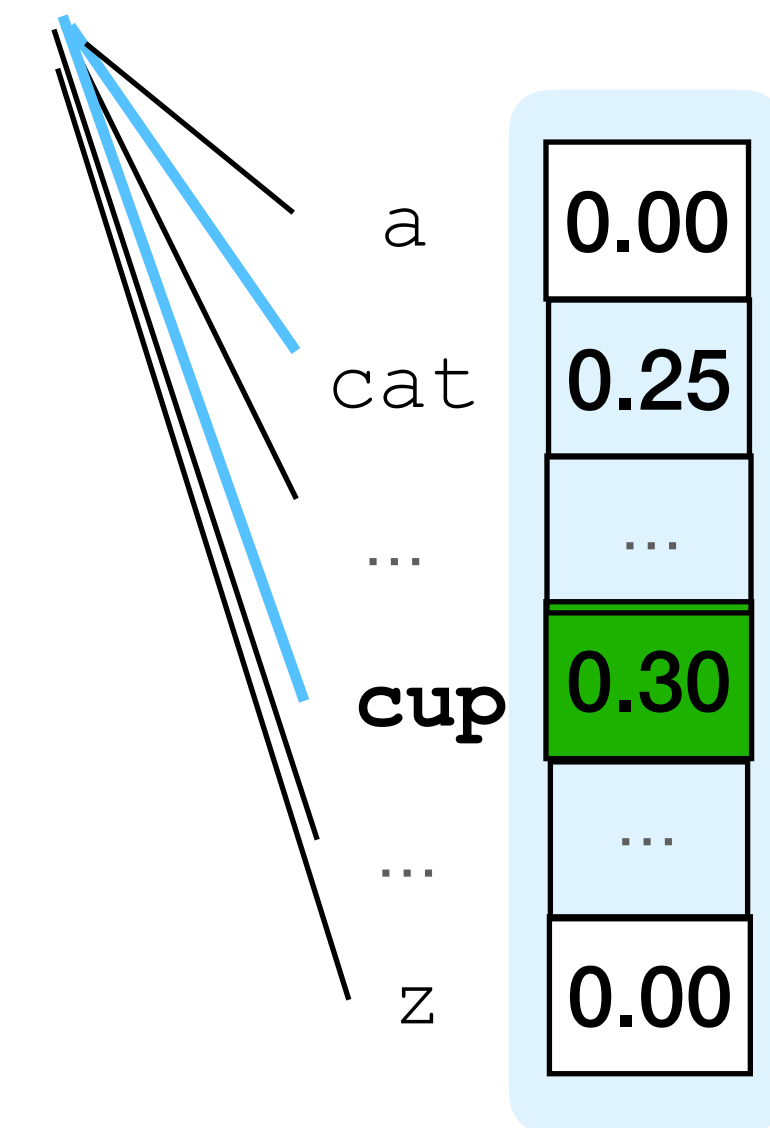
Fluency + constraints-so-far

$$h(\mathbf{y}_{<t+\ell}))$$

Probability of satisfying
Future constraints

Logical Constraints

$(\text{cat} \vee \text{cats}) \wedge (\text{fish})$



NeuroLogic A*esque decoding

- Approximate with a *lookahead heuristic*:

$$\bullet \operatorname{argtopk}_{y_t} \left(s(\mathbf{y}_{\leq t}) + \max_{\text{Lookaheads}} h(\mathbf{y}_{<t+\ell}) \right)$$

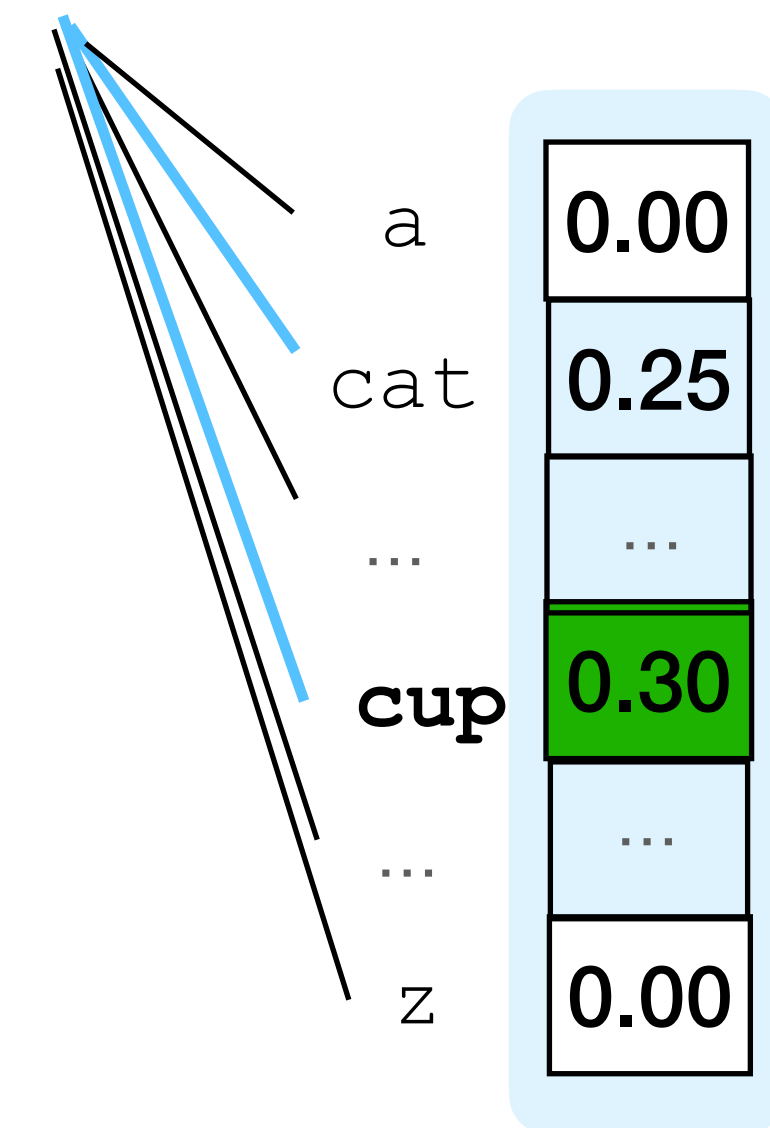
Fluency + constraints-so-far

E.g. single greedy lookahead

Probability of satisfying Future constraints

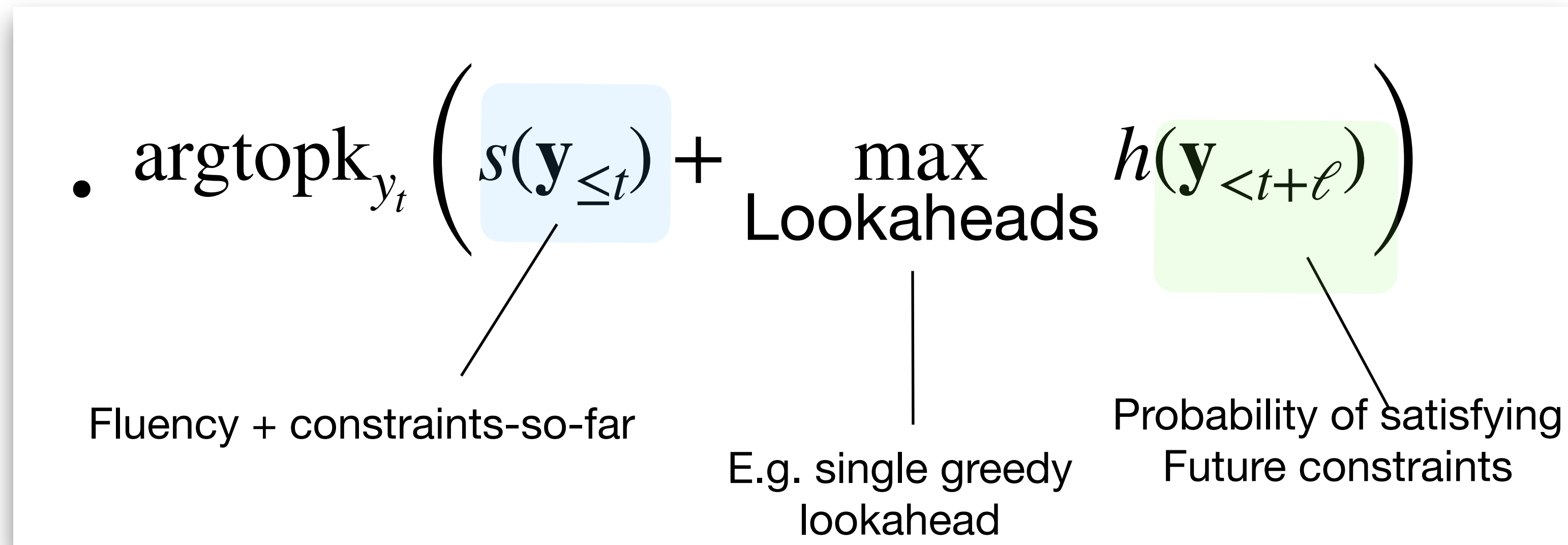
Logical Constraints

$(\text{cat} \vee \text{cats}) \wedge (\text{fish})$

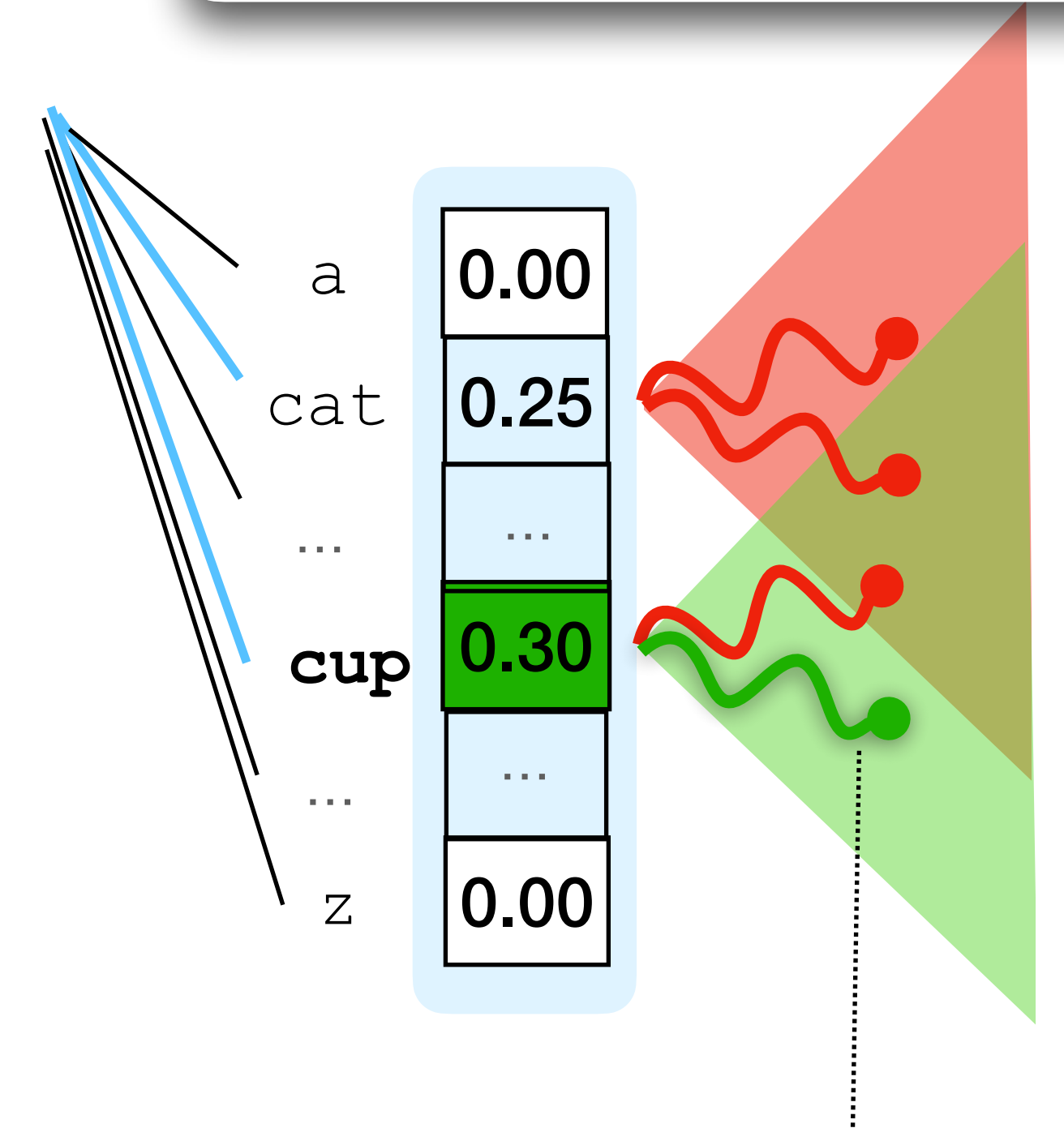


NeuroLogic A*esque decoding

- Approximate with a *lookahead heuristic*:

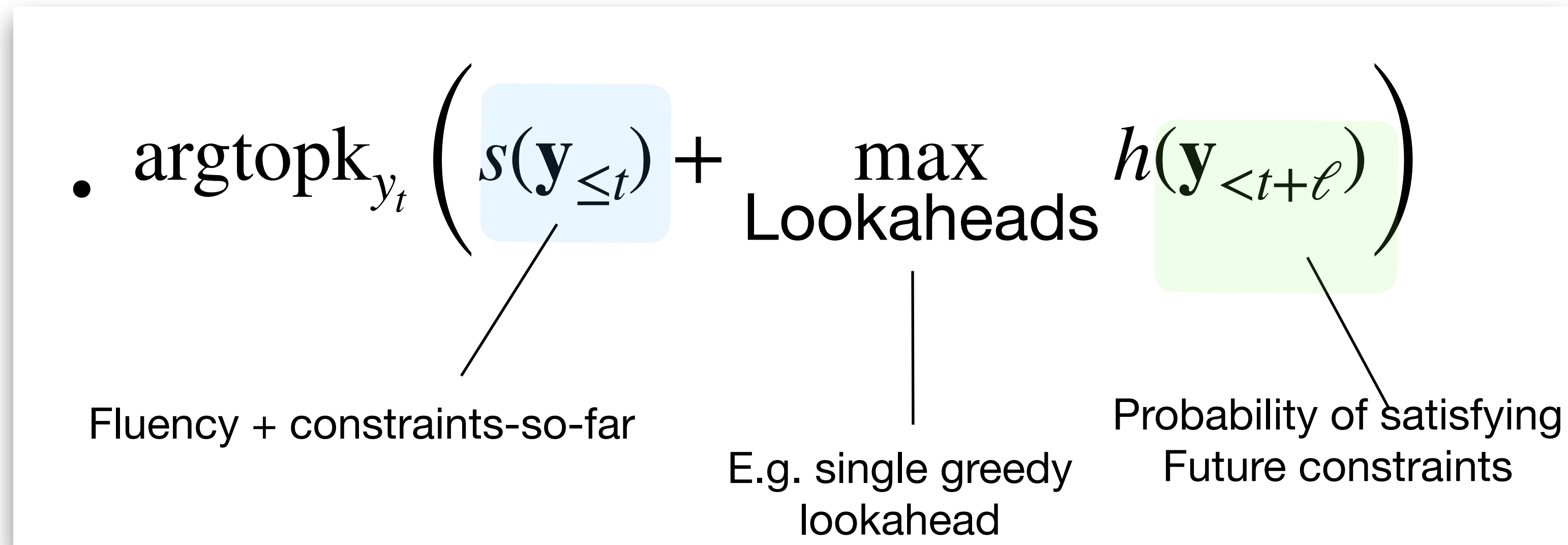


Logical Constraints
 $(\text{cat} \vee \text{cats}) \wedge (\text{fish})$

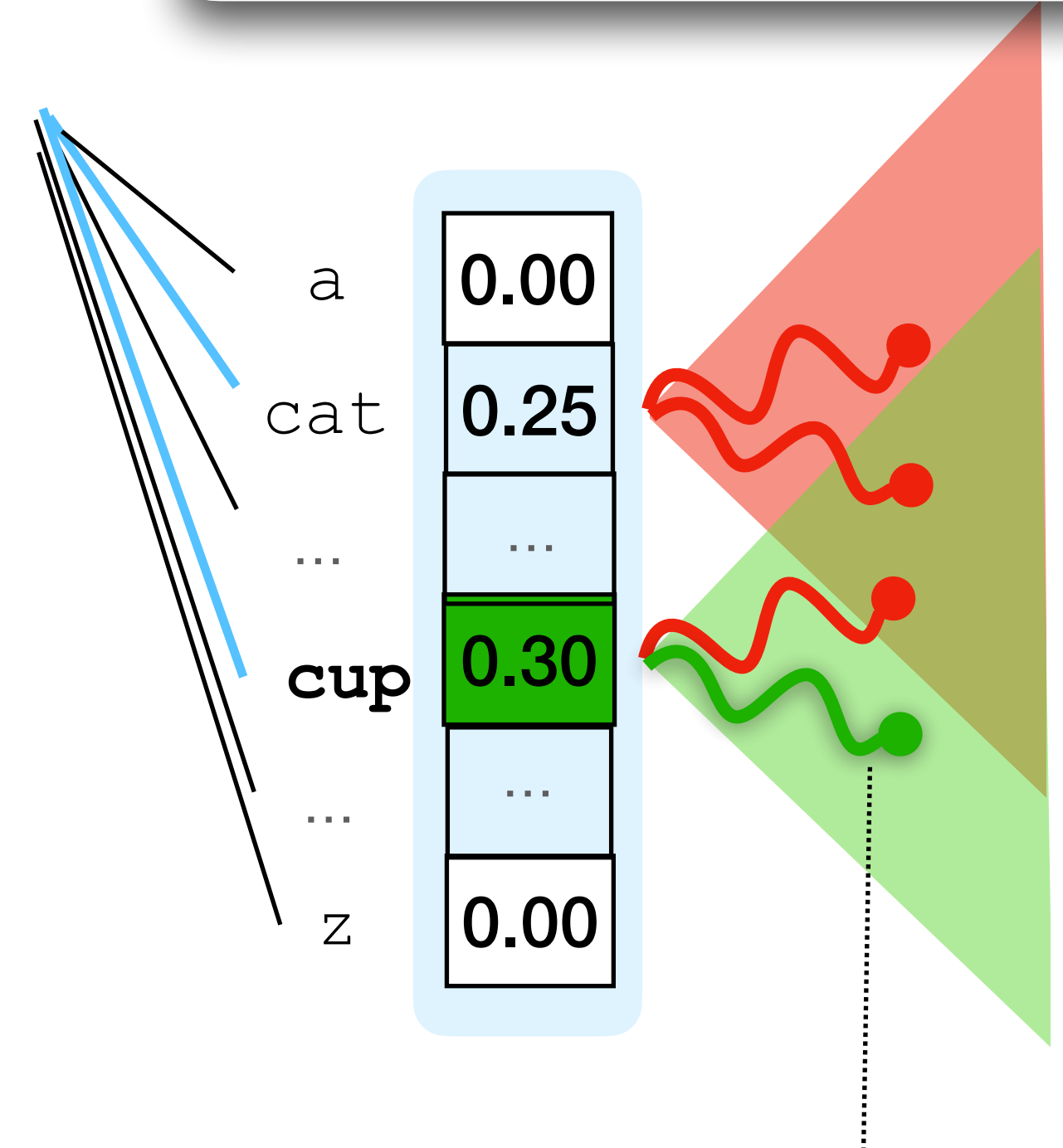


NeuroLogic A*esque decoding

- Approximate with a *lookahead heuristic*:



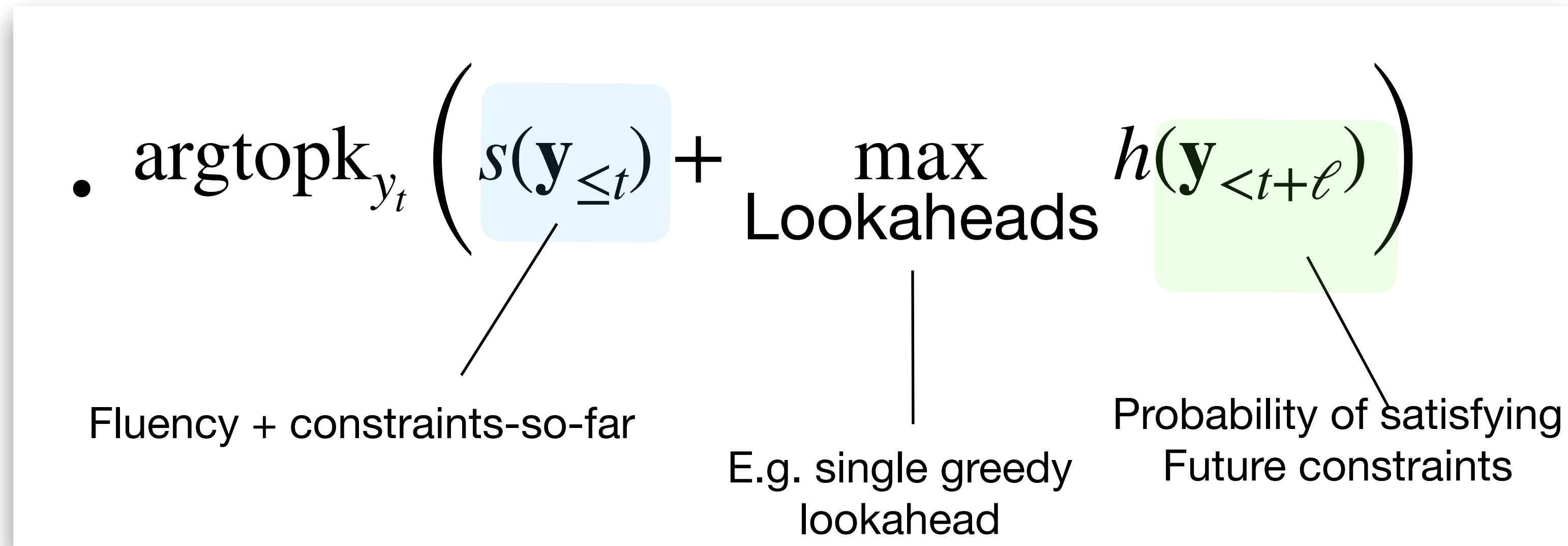
Logical Constraints
 $(\text{cat} \vee \text{cats}) \wedge (\text{fish})$



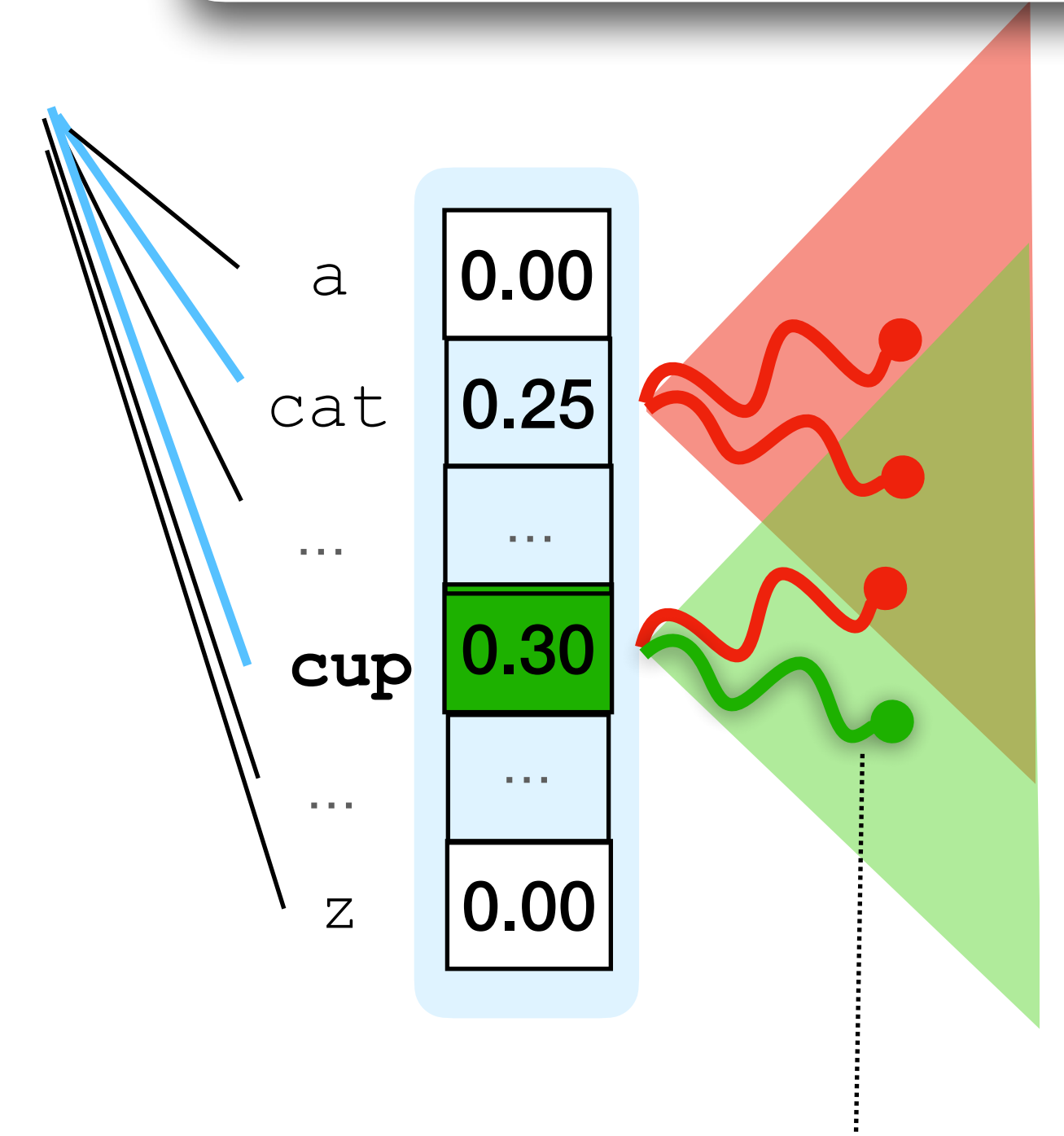
=> my cup has a **fish** and **cat** on it.

NeuroLogic A*esque decoding

- Approximate with a *lookahead heuristic*:



Logical Constraints
 $(\text{cat} \vee \text{cats}) \wedge (\text{fish})$

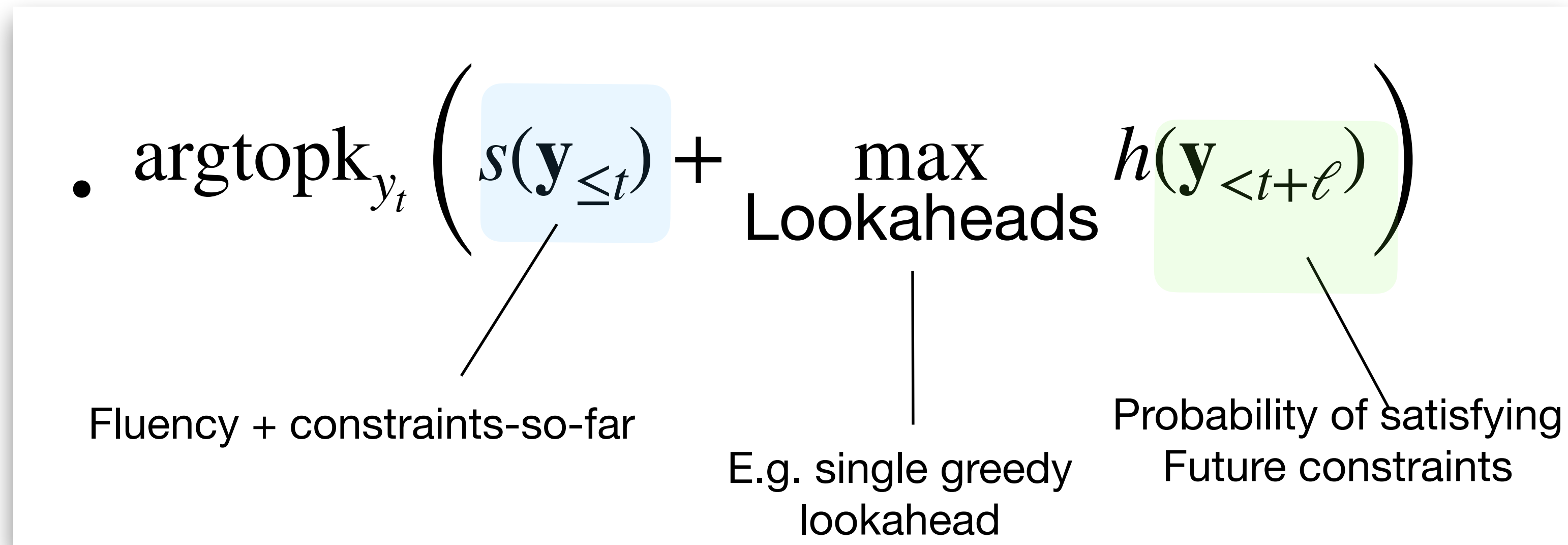


=> my cup has a **fish** and **cat** on it.

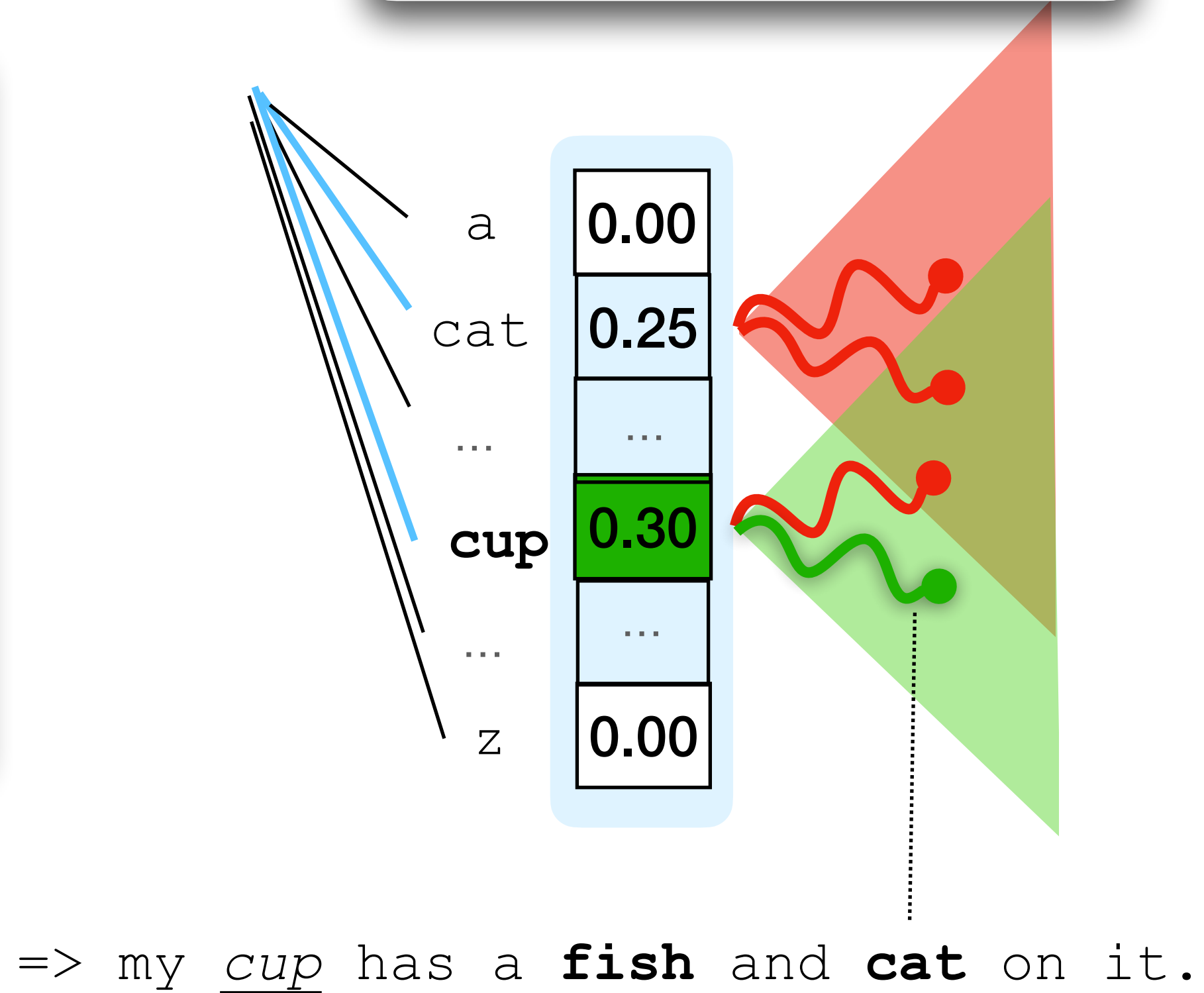


NeuroLogic A*esque decoding

- Approximate with a *lookahead heuristic*:



Logical Constraints
 $(\text{cat} \vee \text{cats}) \wedge (\text{fish})$



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



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.

(sponge ∨ sponges) ∧ (pour ∨
pours ∨ pouring ∨ poured) ∧
(pool ∨ pools) ∧ (side ∨ sides) ∧
(clean ∨ clean ∨ cleans ∨ cleaning)

C

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

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.

$(\text{sponge} \vee \text{sponges}) \wedge (\text{pour} \vee \text{pours} \vee \text{pouring} \vee \text{poured}) \wedge$
 $(\text{pool} \vee \text{pools}) \wedge (\text{side} \vee \text{sides}) \wedge$
 $(\text{clean} \vee \text{clean} \vee \text{cleans} \vee \text{cleaning})$

C

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

A* NeuroLogic

The boy **cleaned** the **side** of the **pool** with a **sponge**, and **poured** water over it .

Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

Off-the-shelf GPT-2

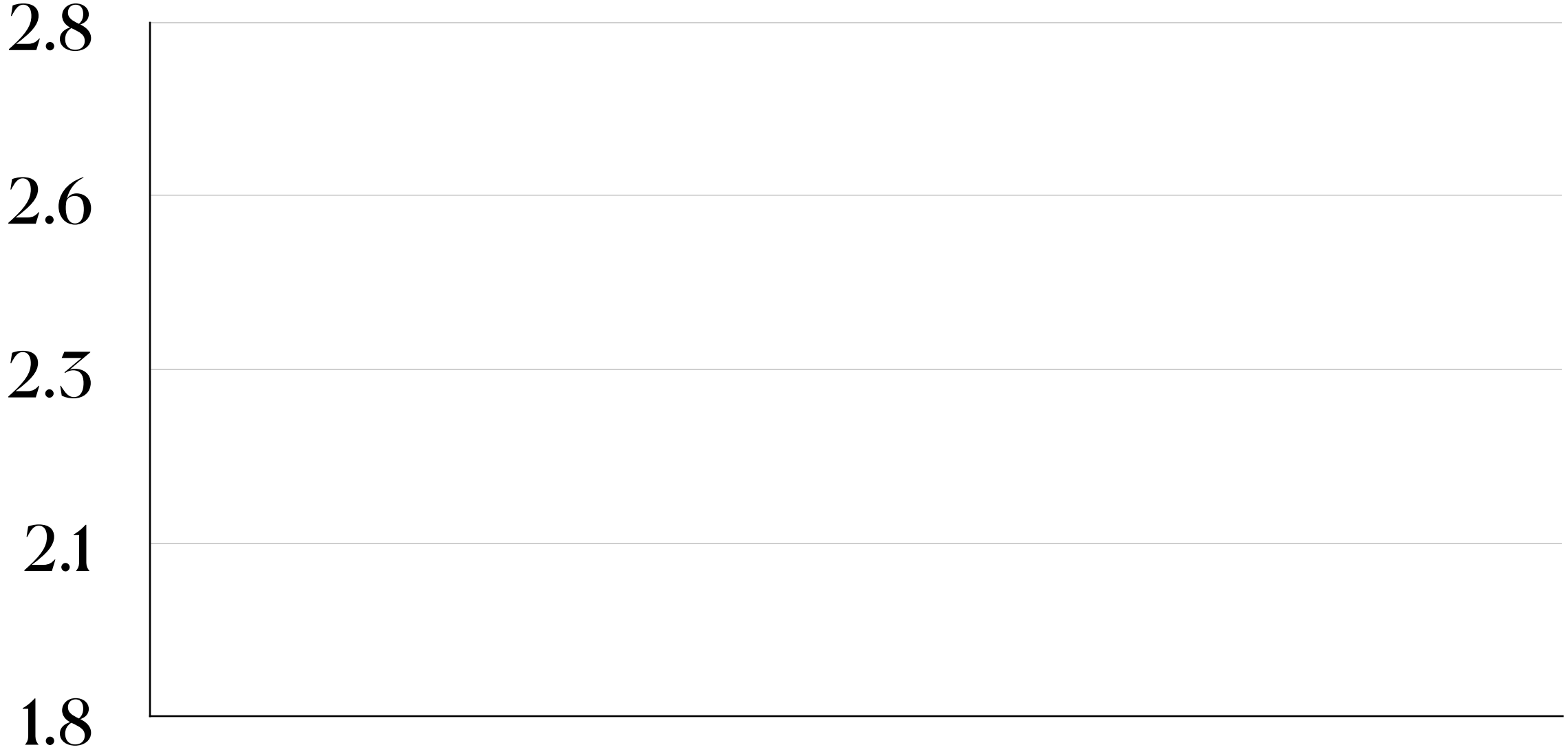
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

Off-the-shelf GPT-2

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



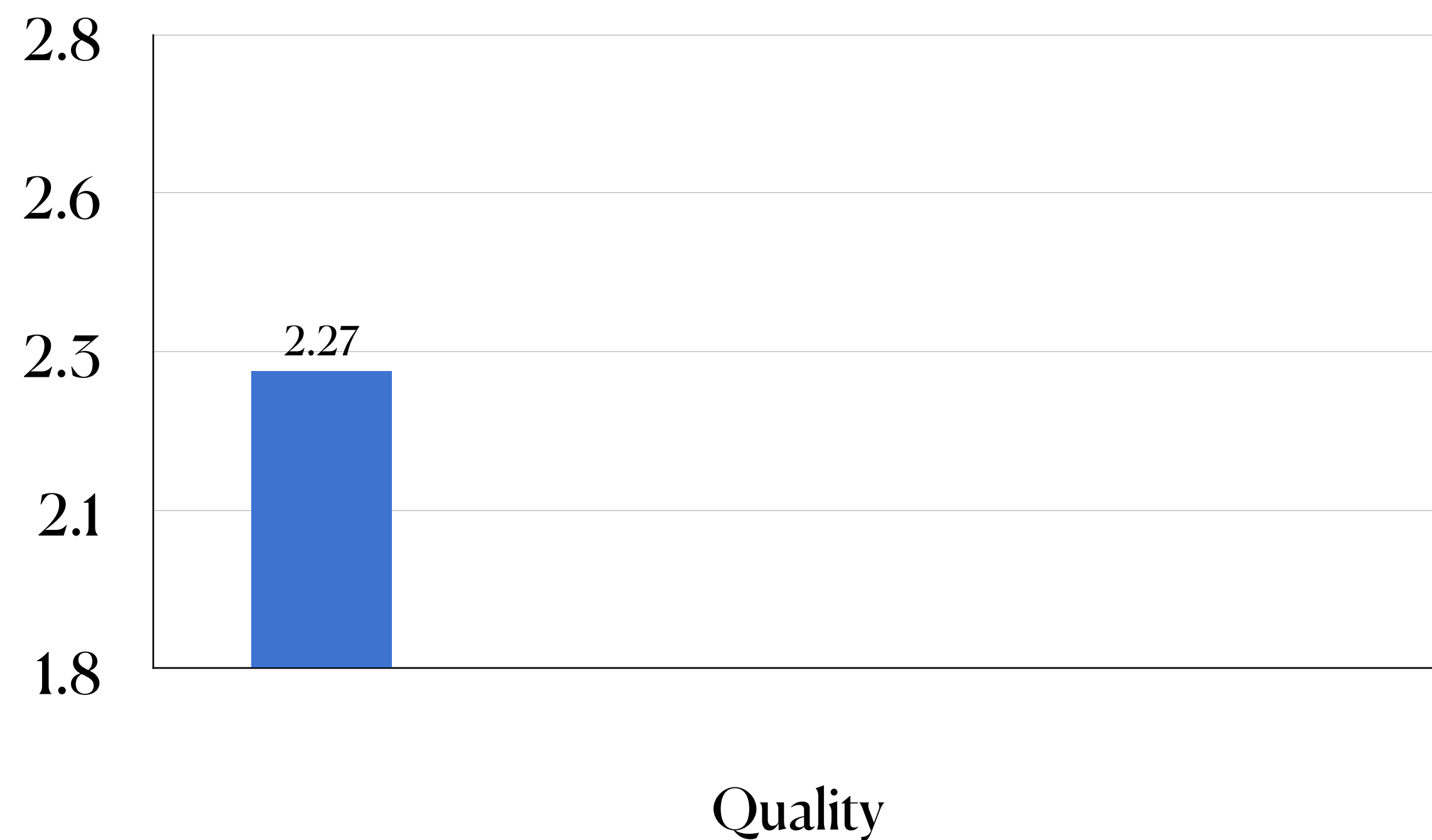
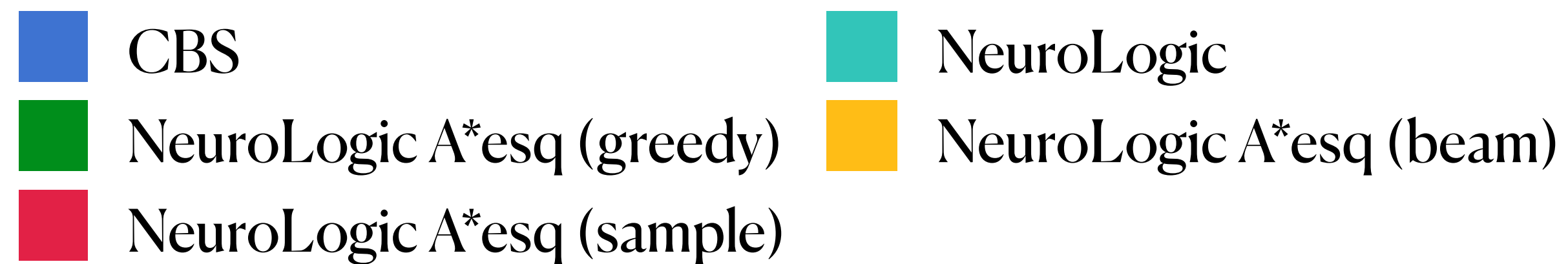
Quality

Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

Off-the-shelf GPT-2

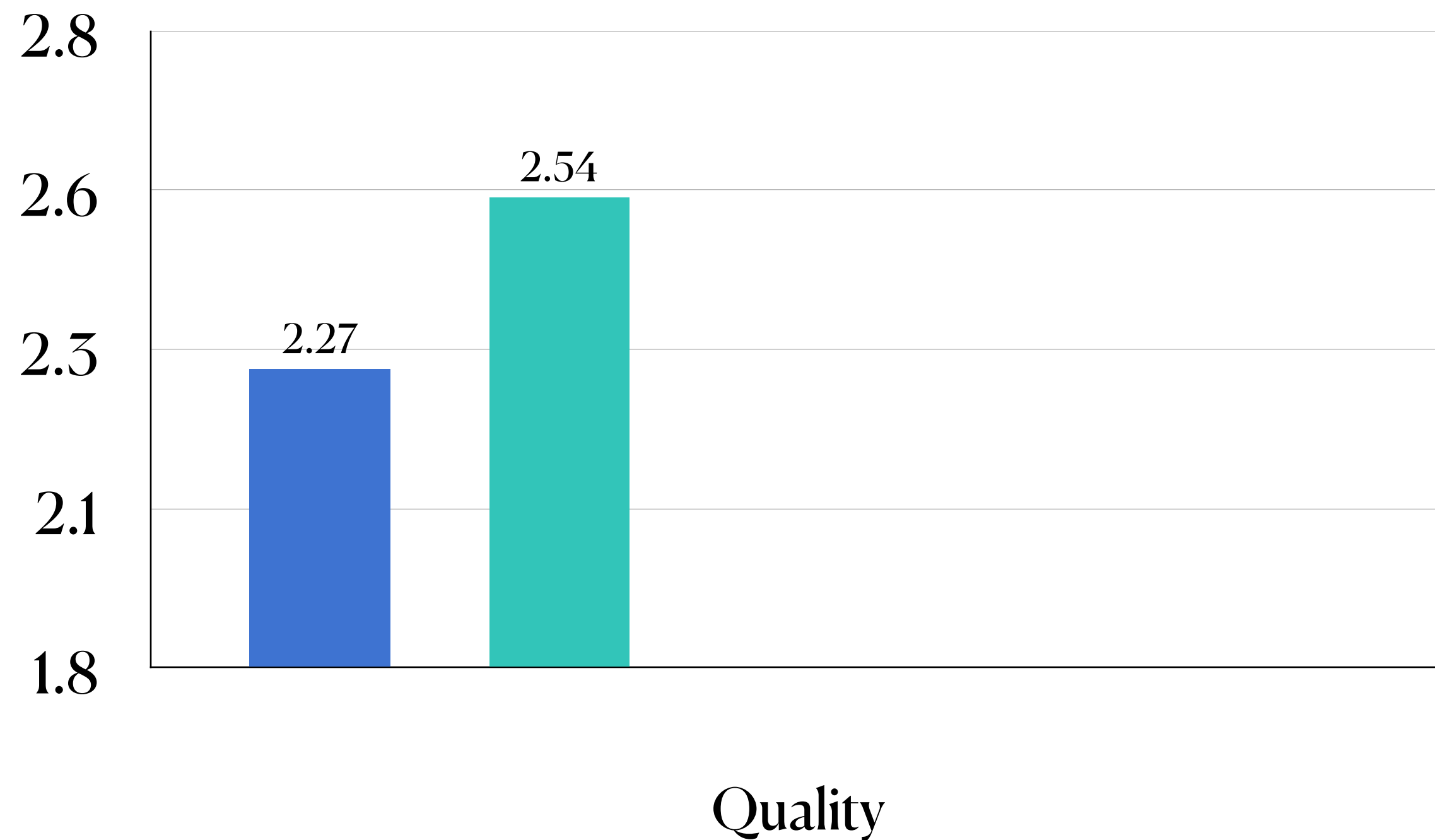
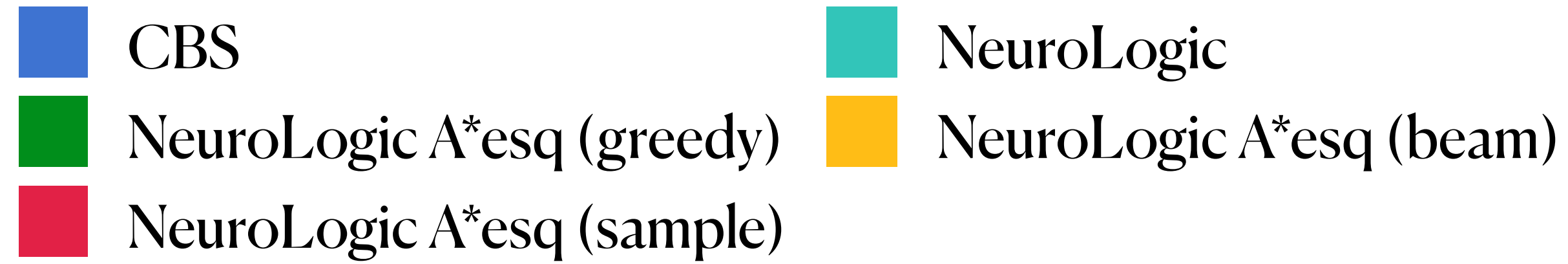


Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

Off-the-shelf GPT-2

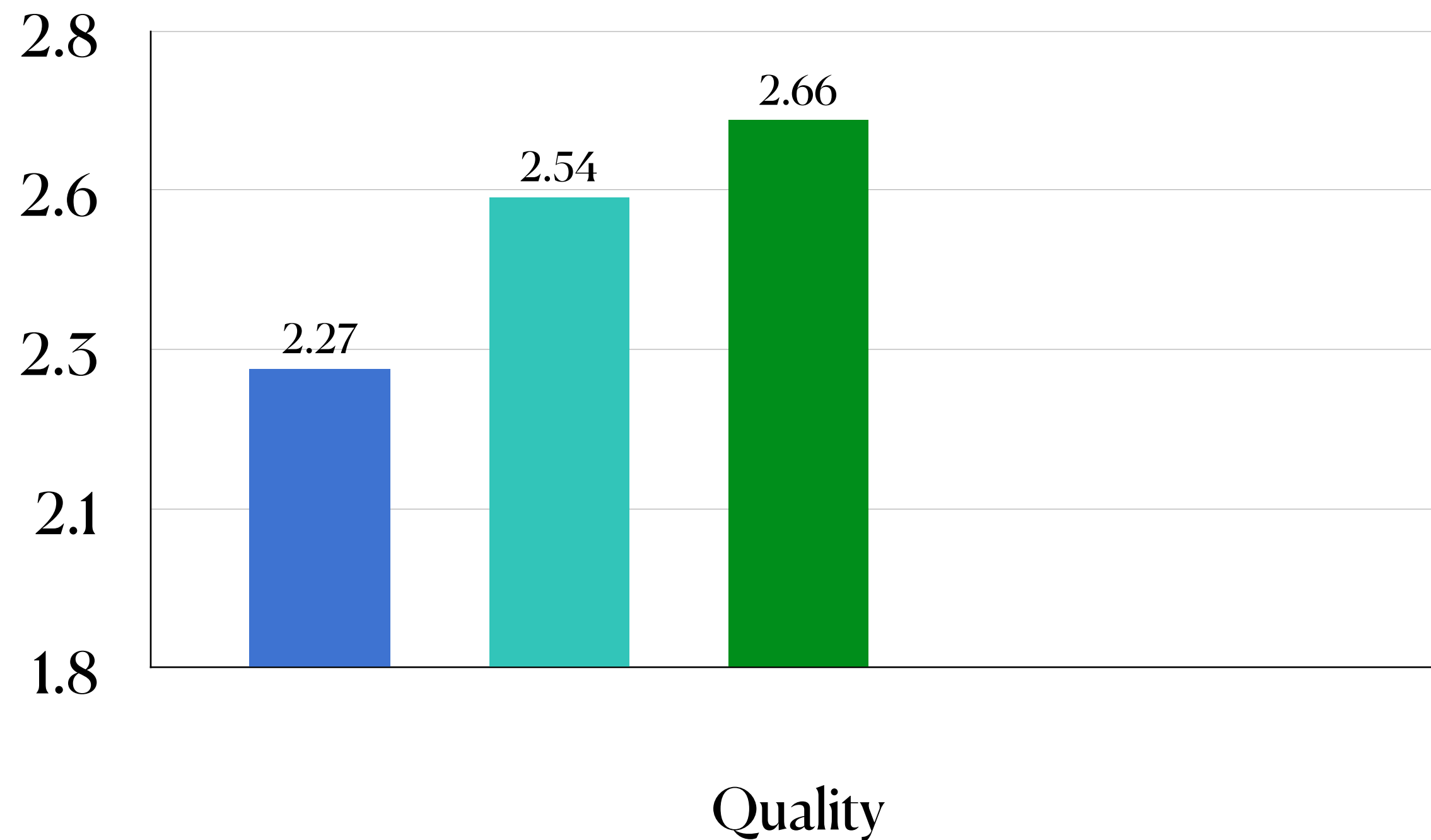
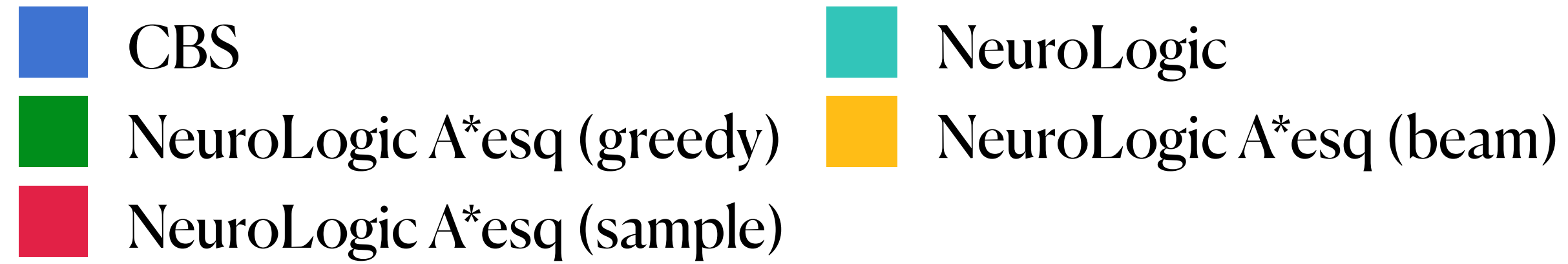


Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

Off-the-shelf GPT-2

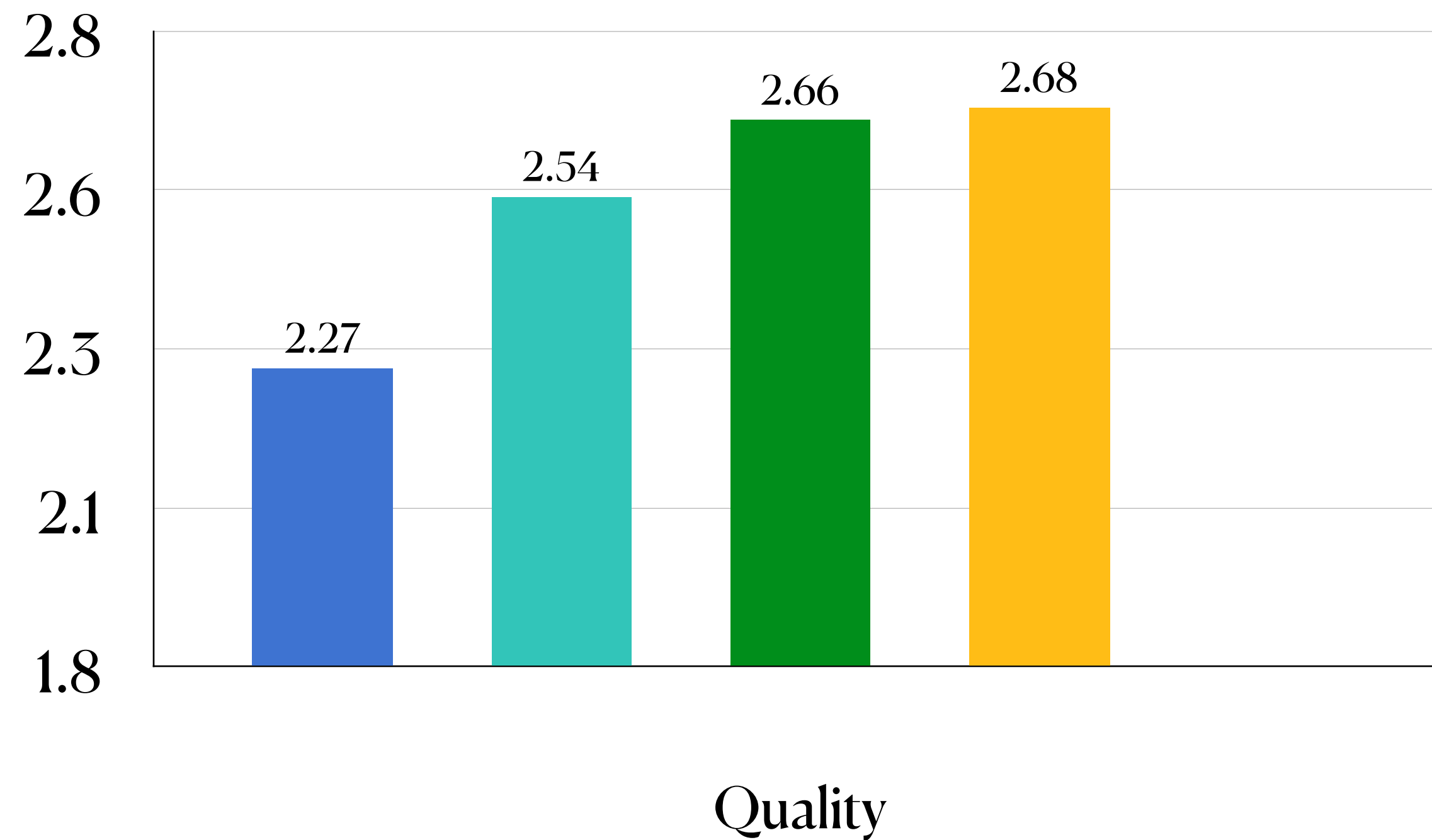
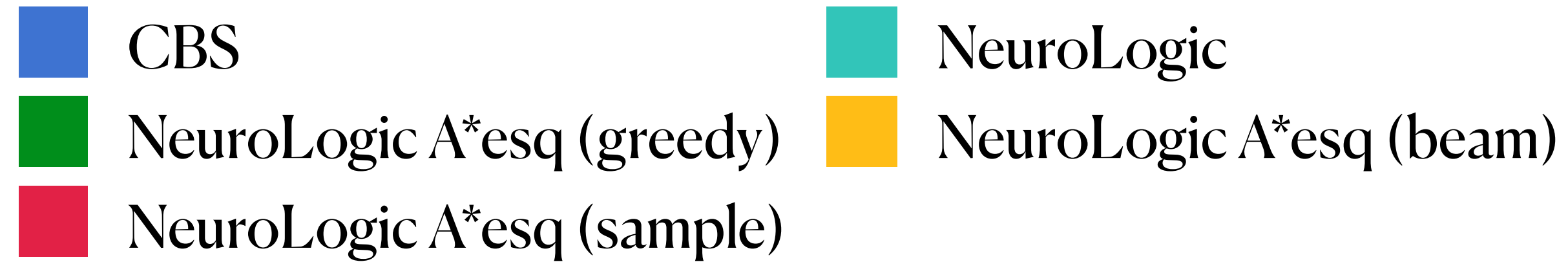


Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

Off-the-shelf GPT-2

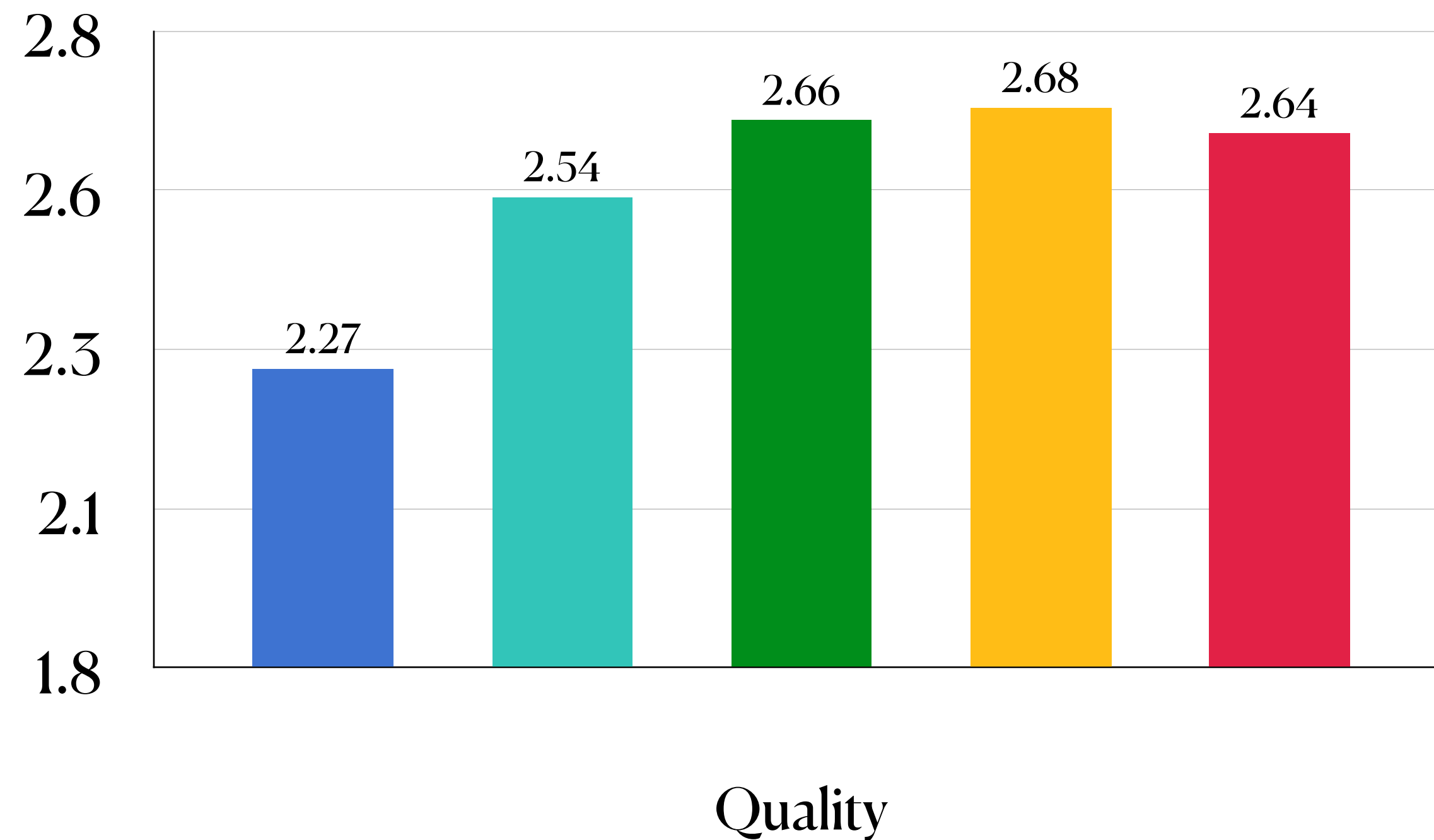
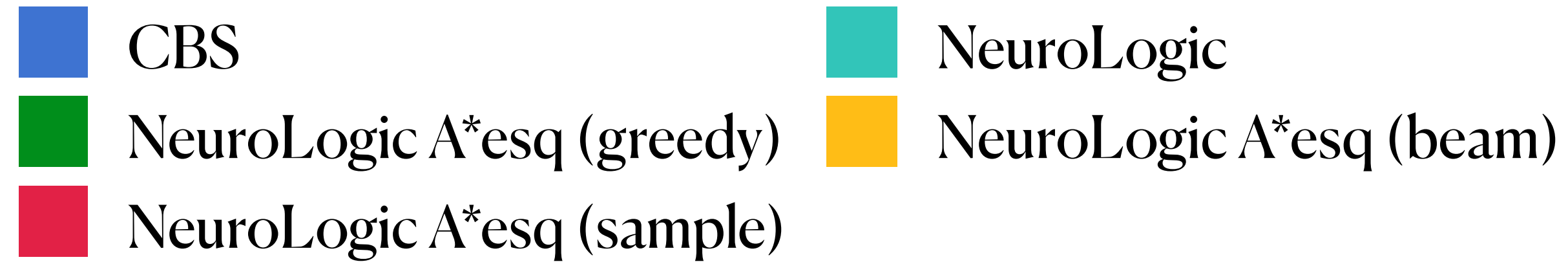


Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

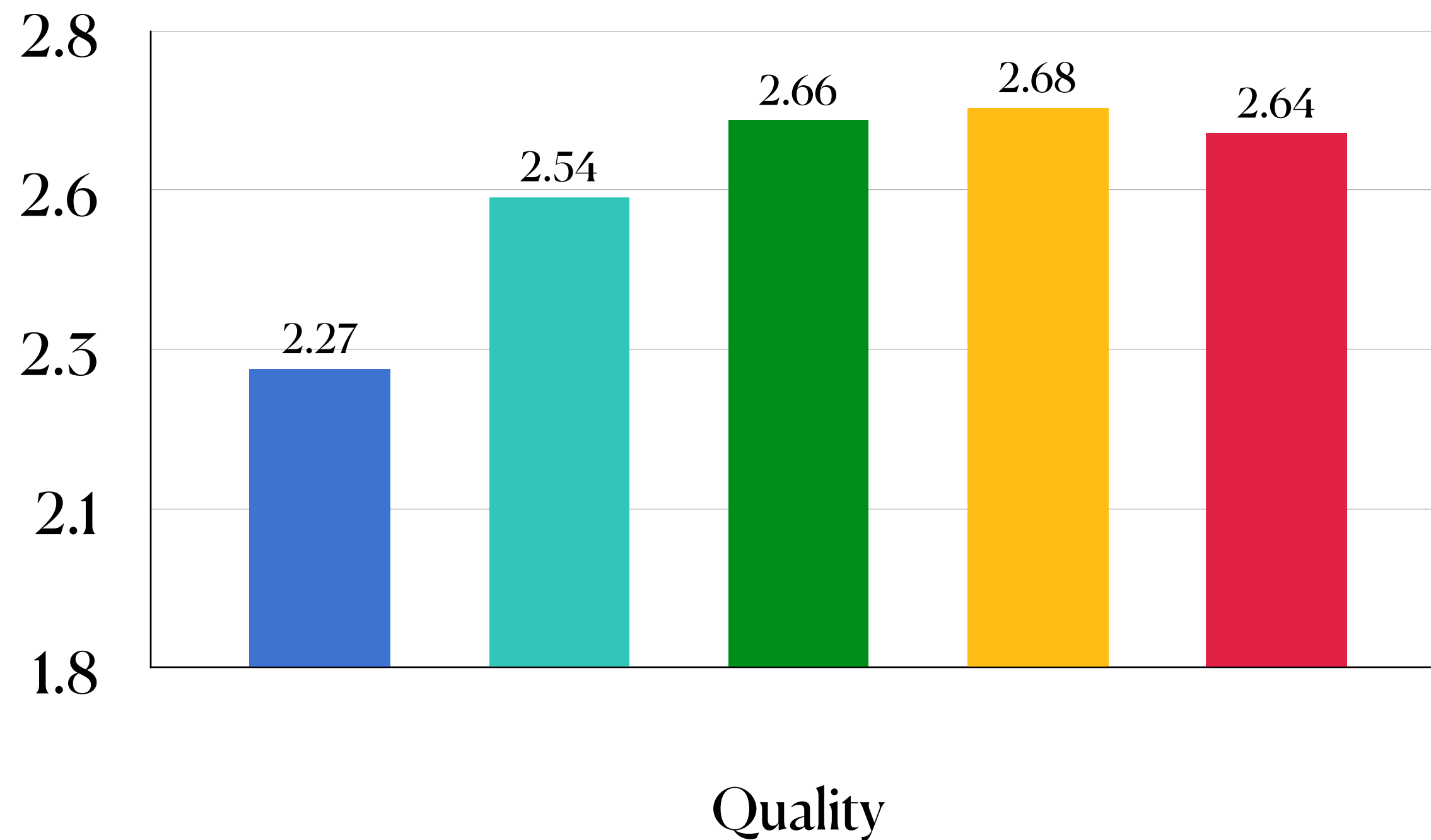
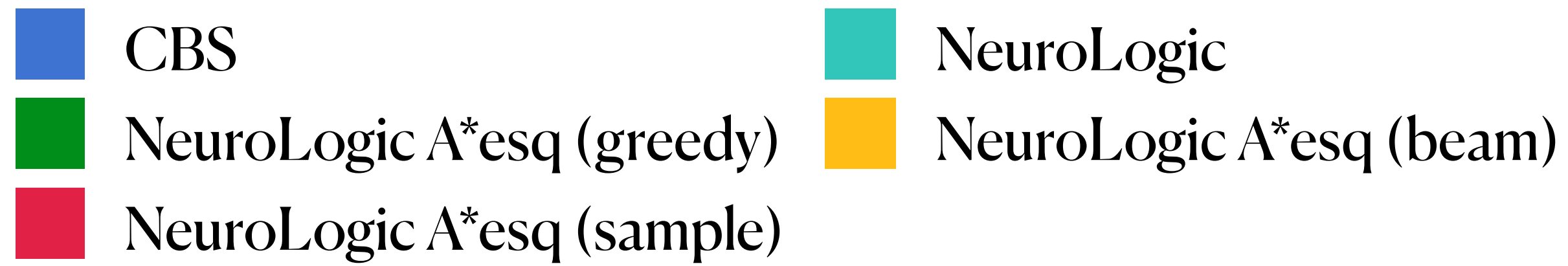
Off-the-shelf GPT-2



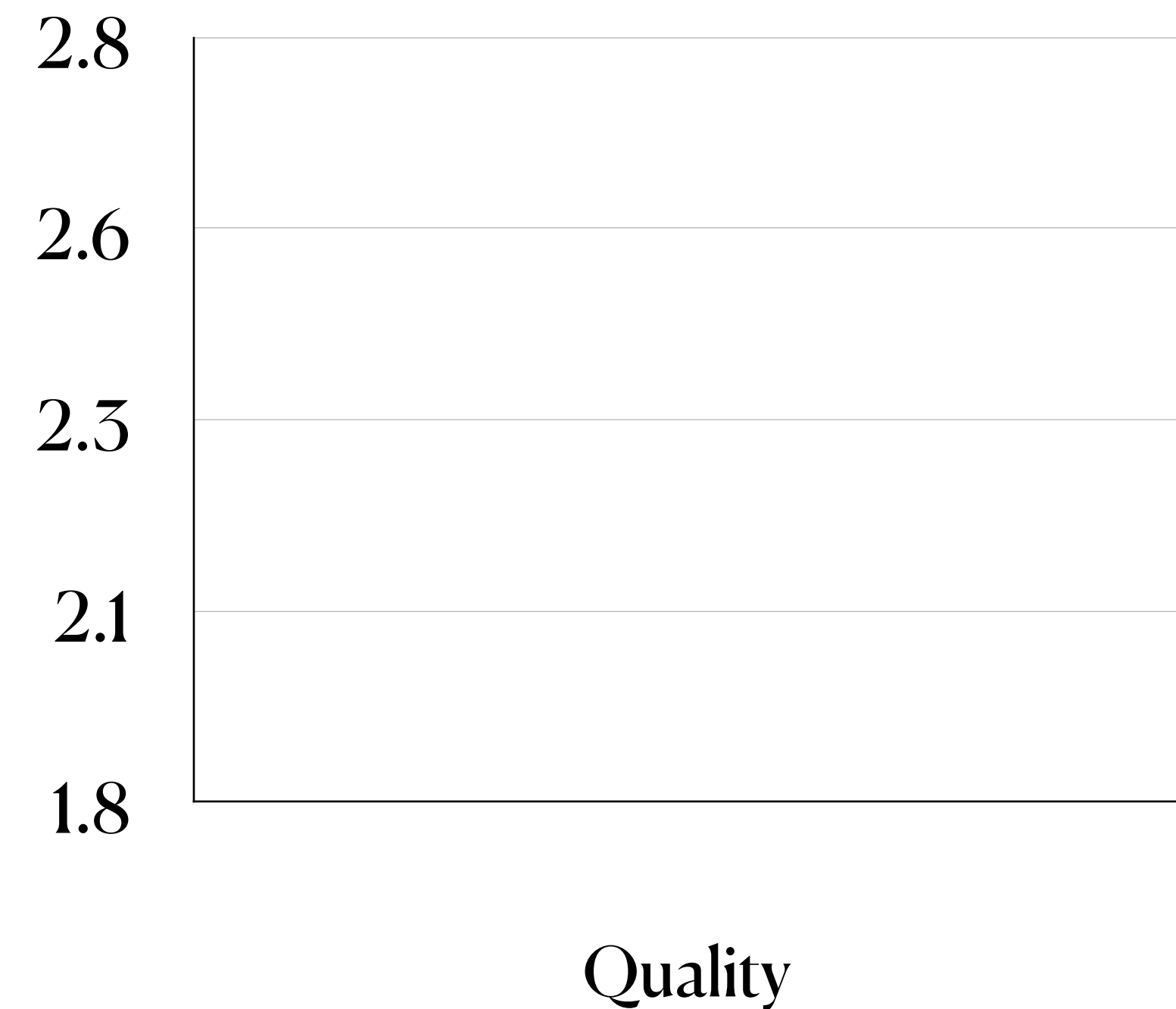
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2



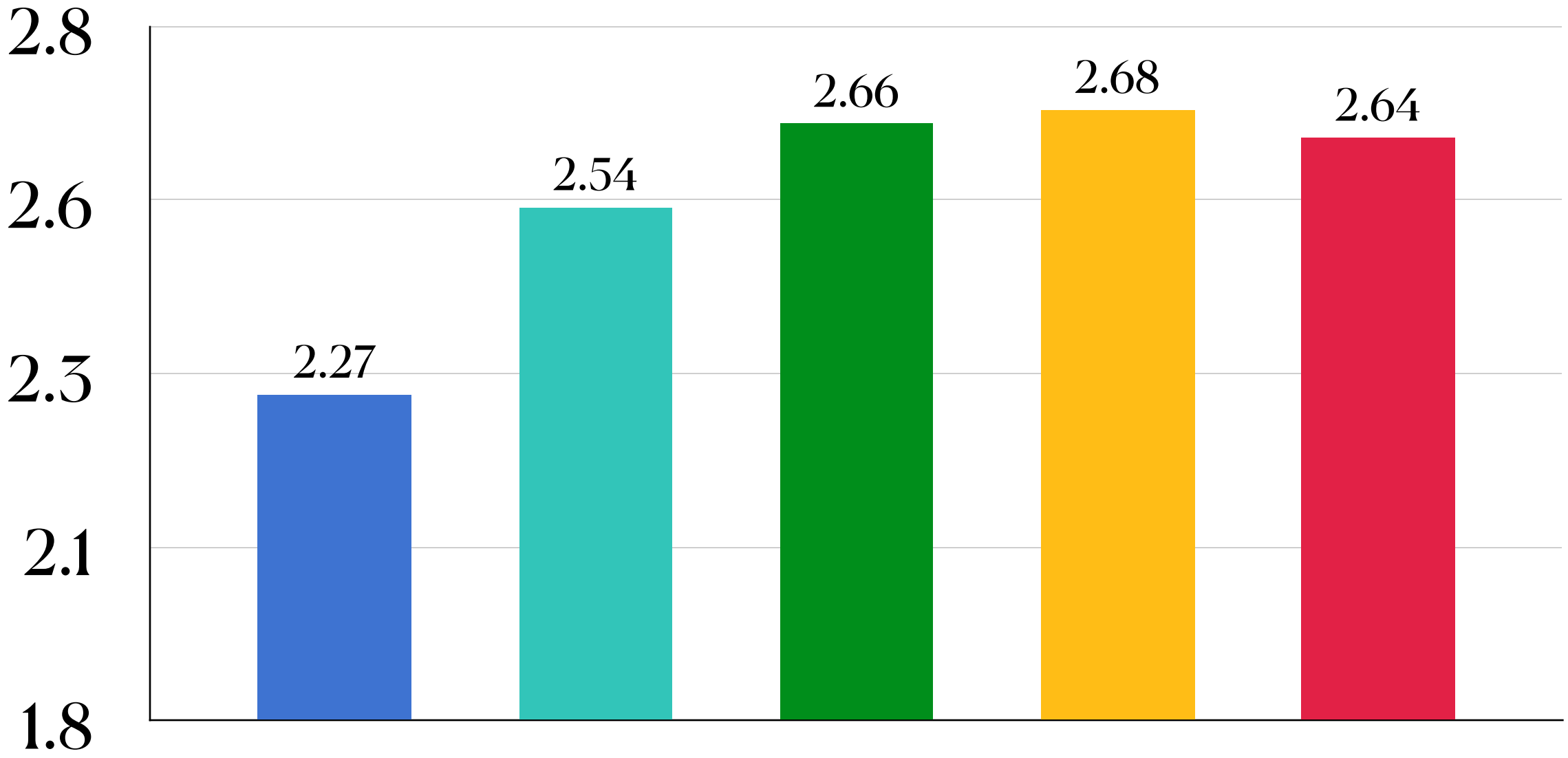
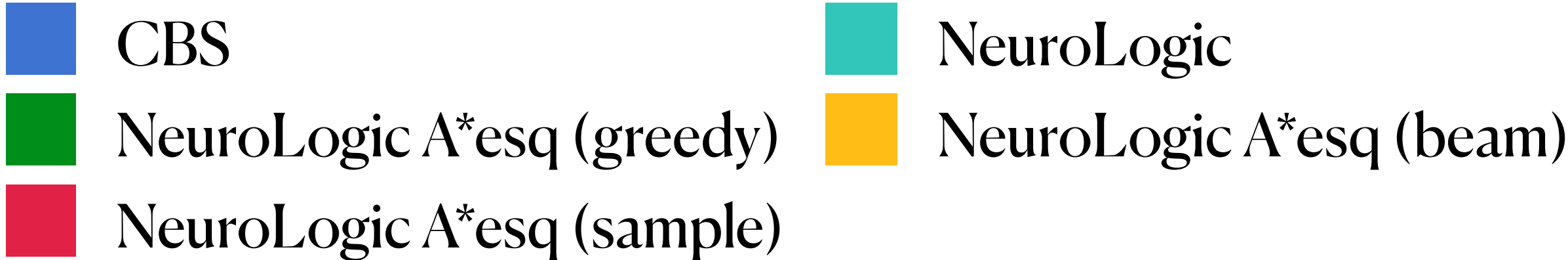
Off-the-shelf GPT-2



Human evaluation | CommonGen

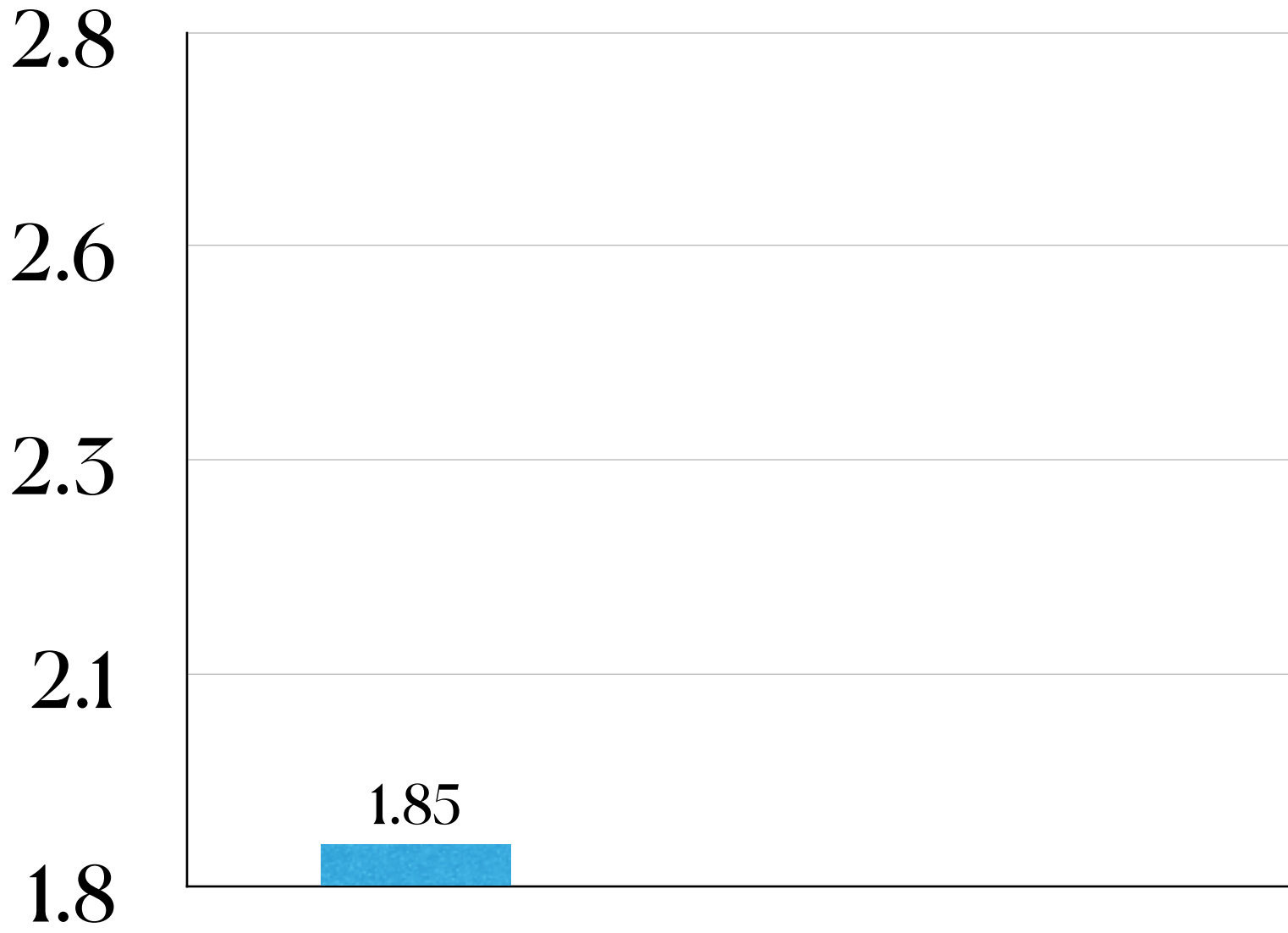
(Lin et al., 2020)

Fine-tuned GPT-2



Quality

Off-the-shelf GPT-2



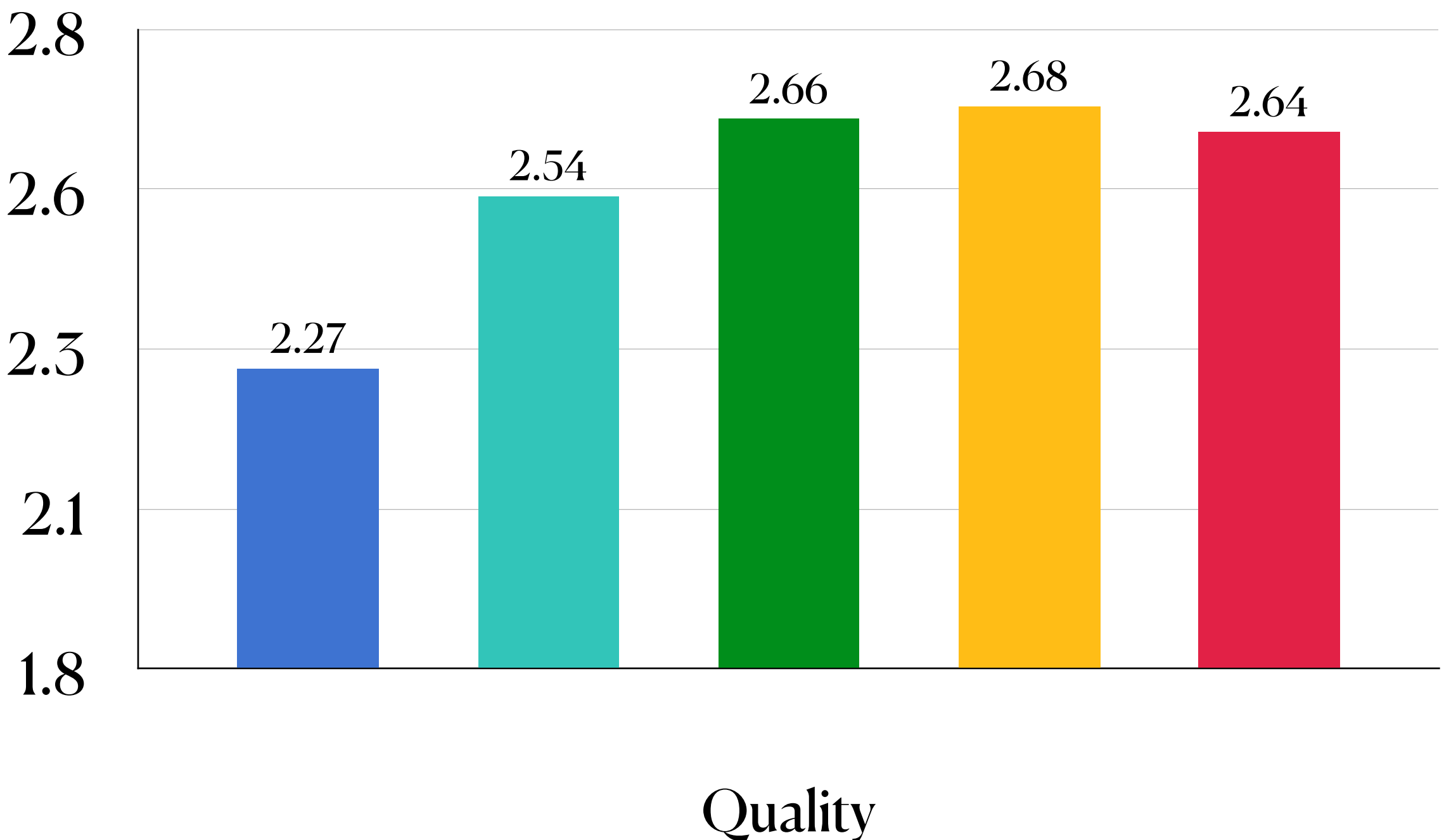
Quality

Human evaluation | CommonGen

(Lin et al., 2020)

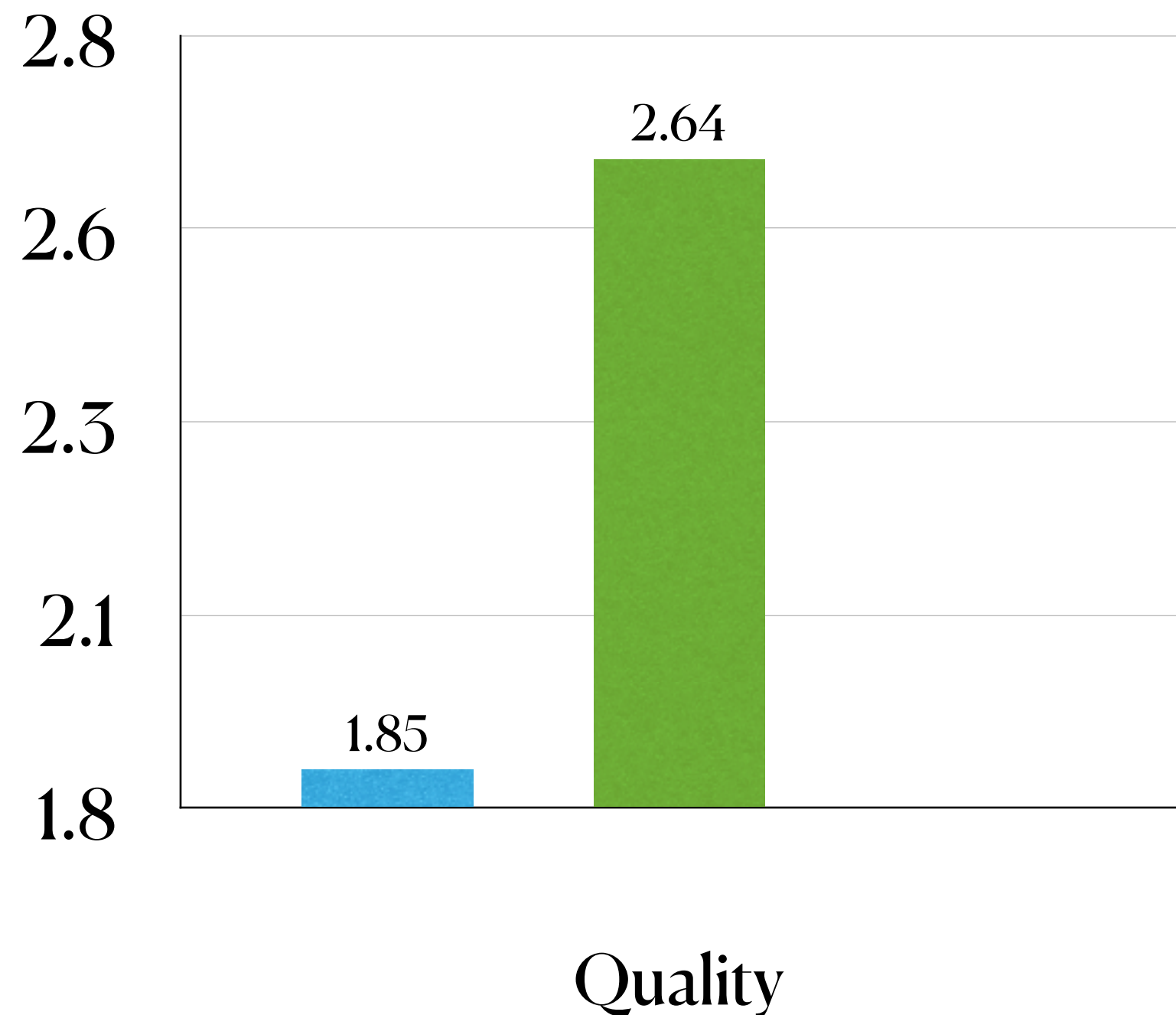
Fine-tuned GPT-2

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



Off-the-shelf GPT-2

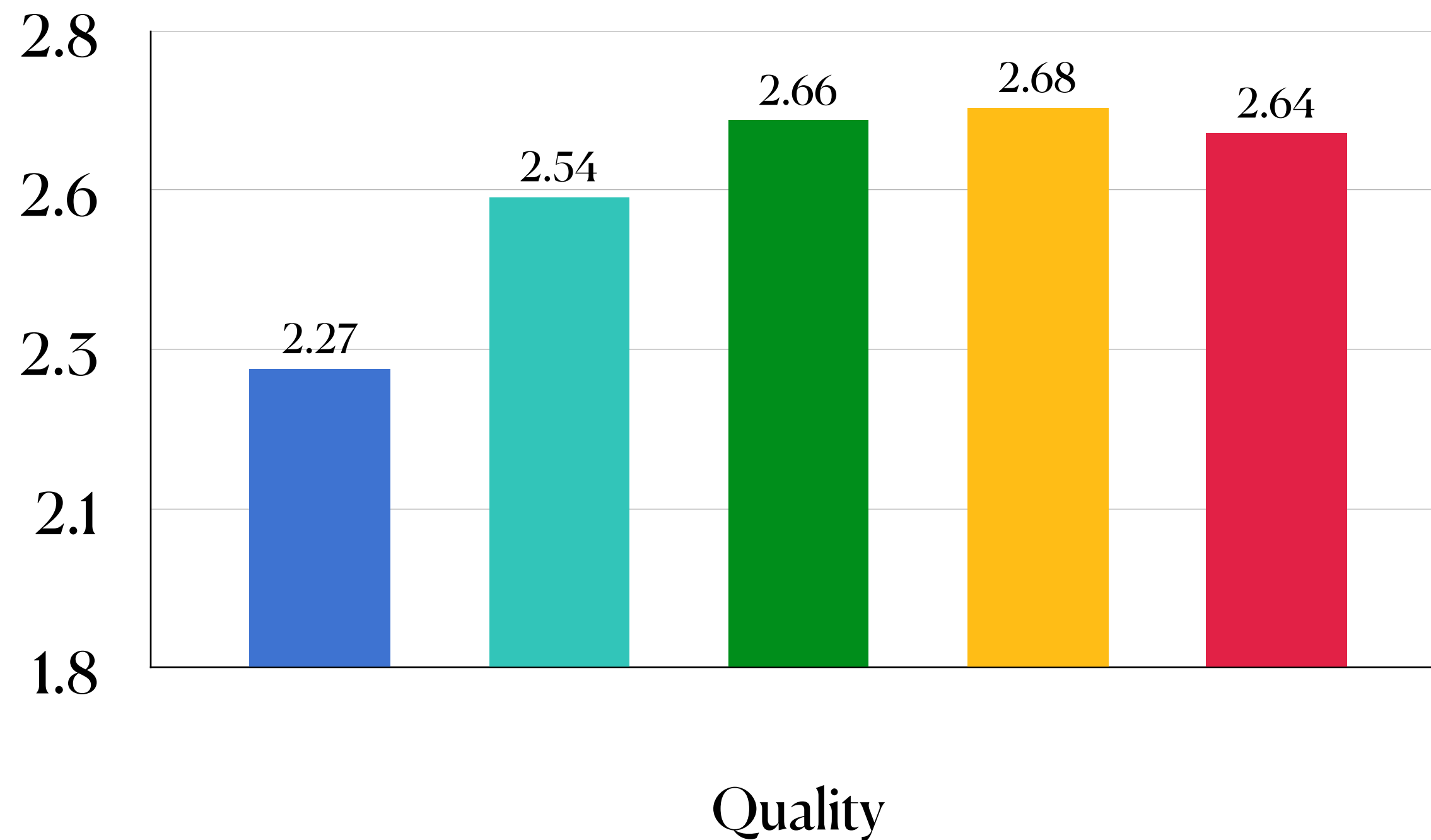
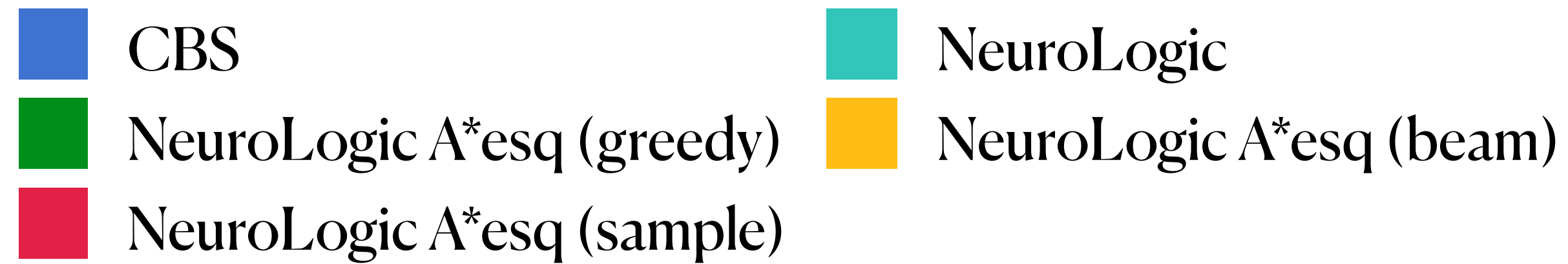
- TSMH
- NeuroLogic
- NeuroLogic A*esq (greedy)



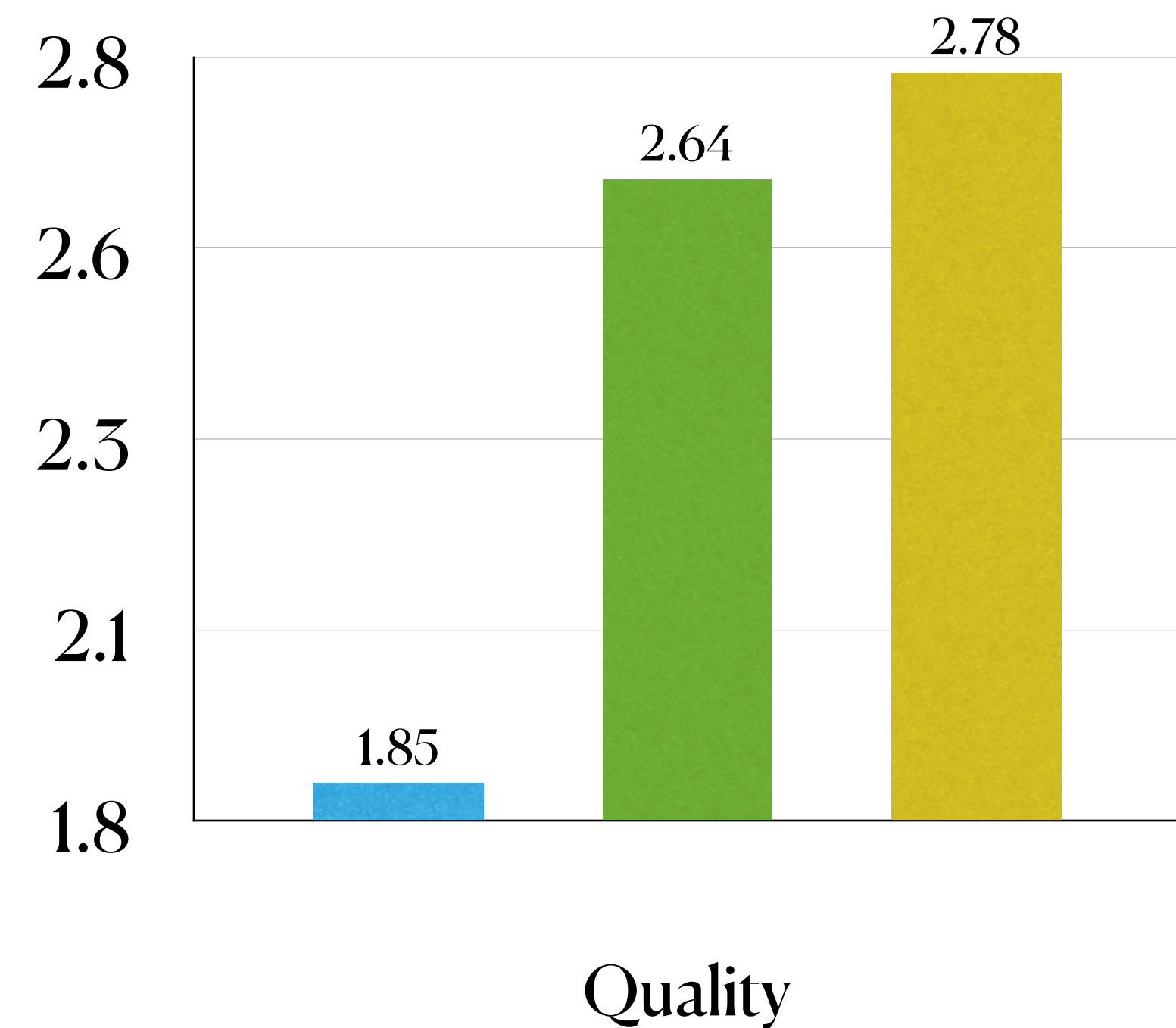
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2



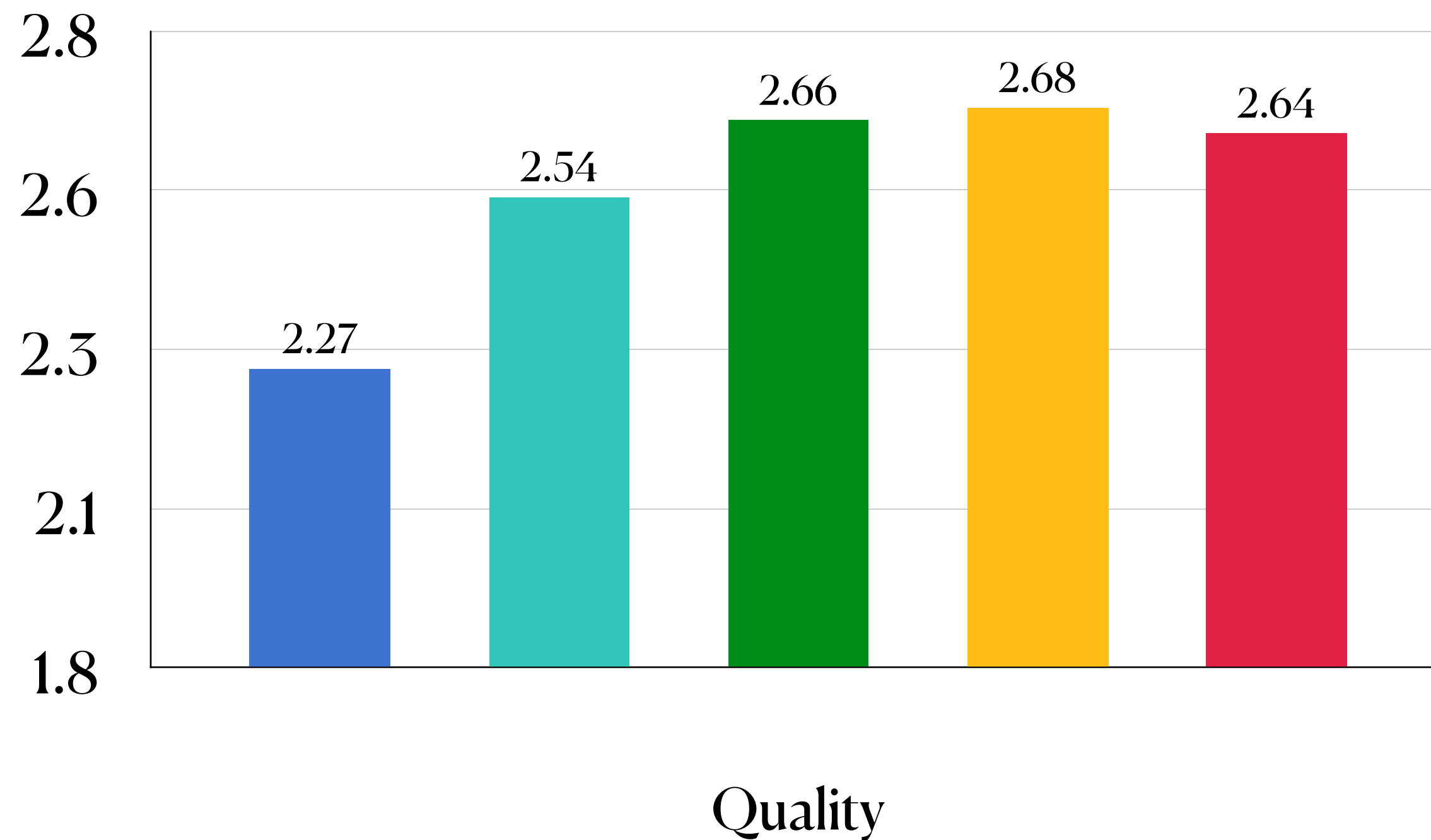
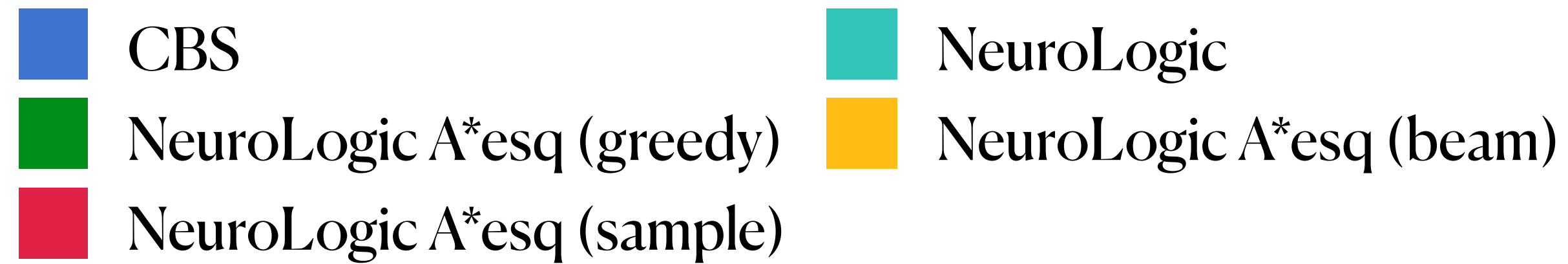
Off-the-shelf GPT-2



Human evaluation | CommonGen

(Lin et al., 2020)

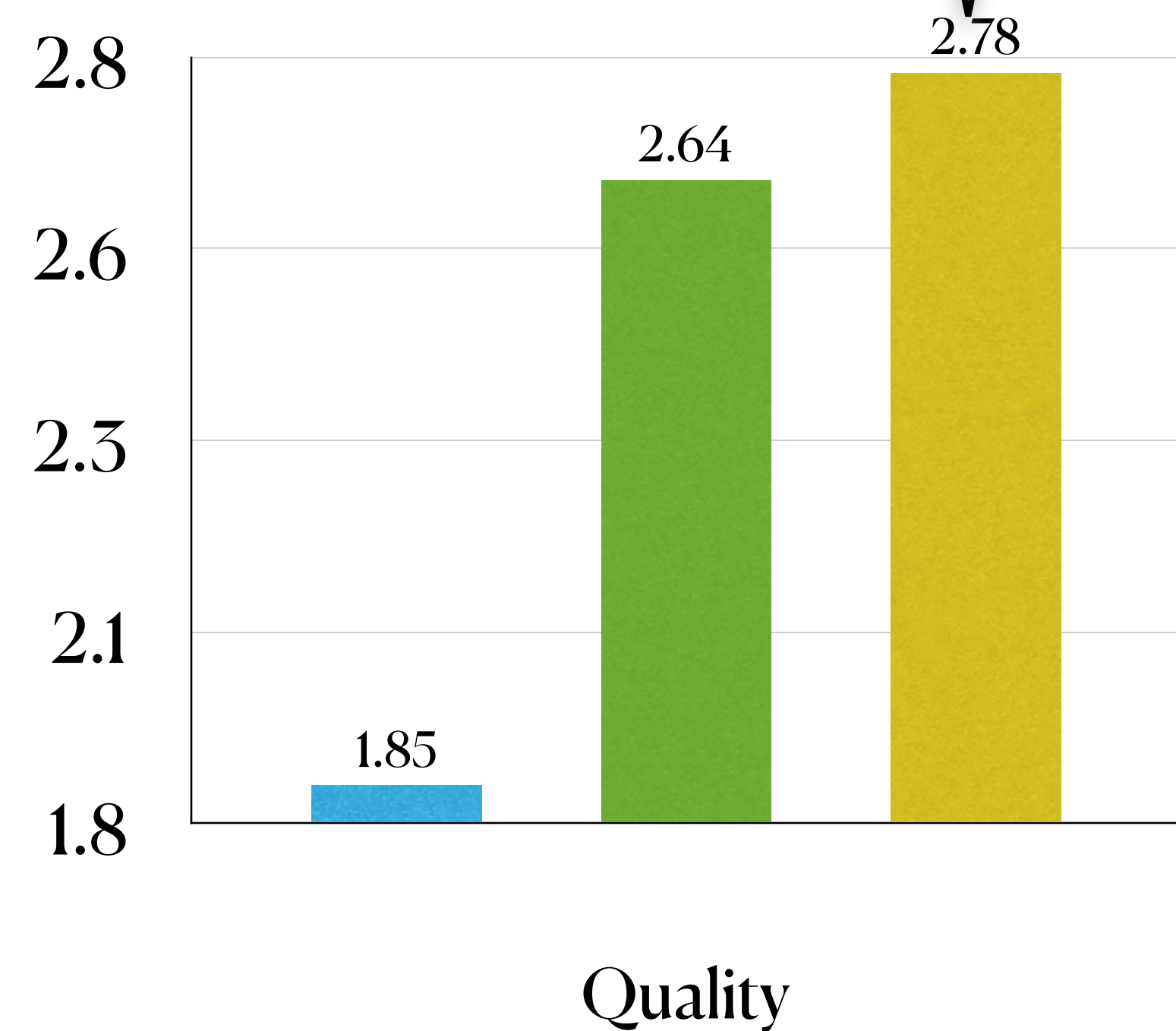
Fine-tuned GPT-2



Off-the-shelf GPT-2



Off-the-shelf A* outperforms all fine-tuned methods



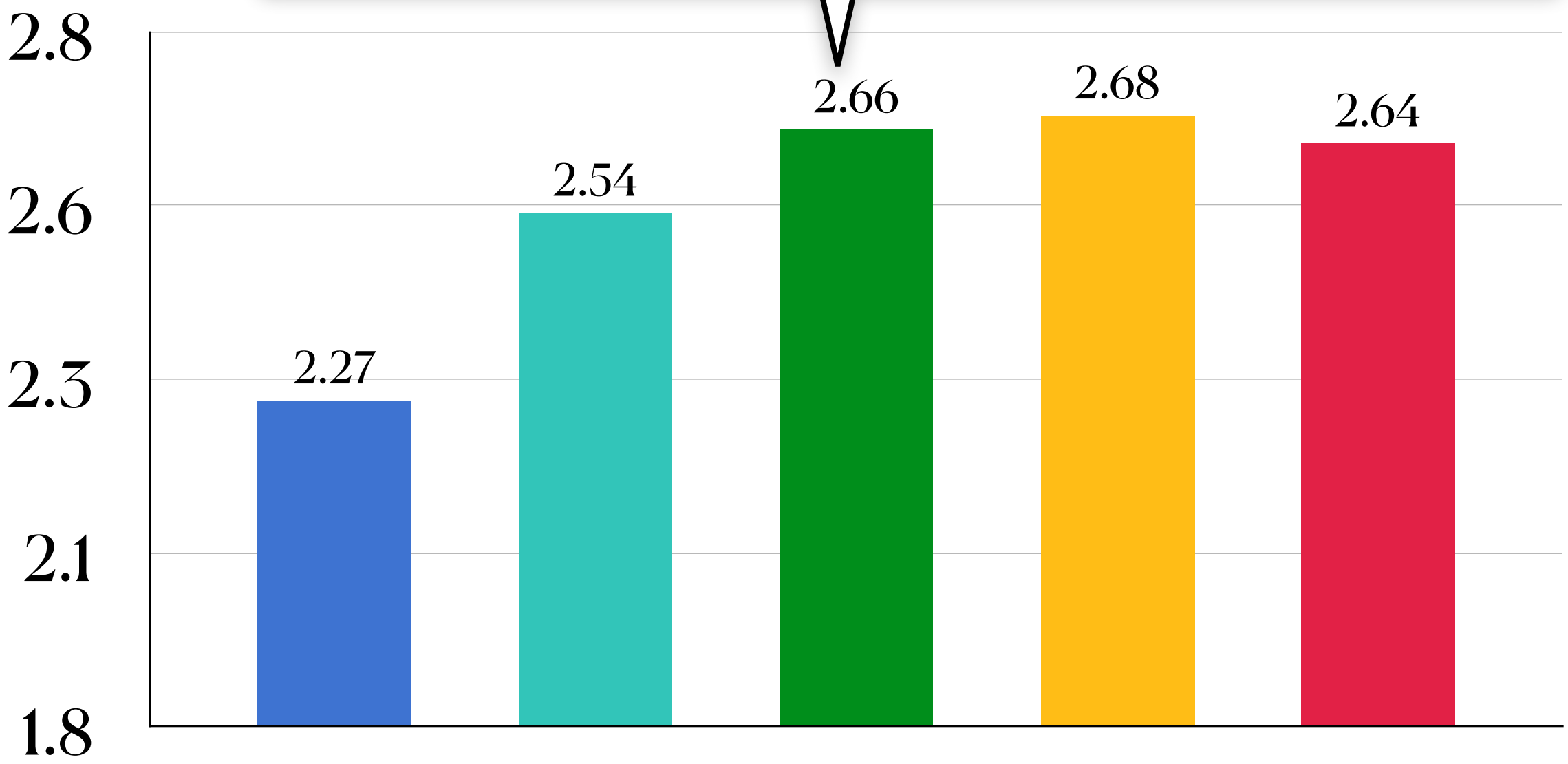
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

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

A* NeuroLogic with greedy lookahead:
efficient & performant

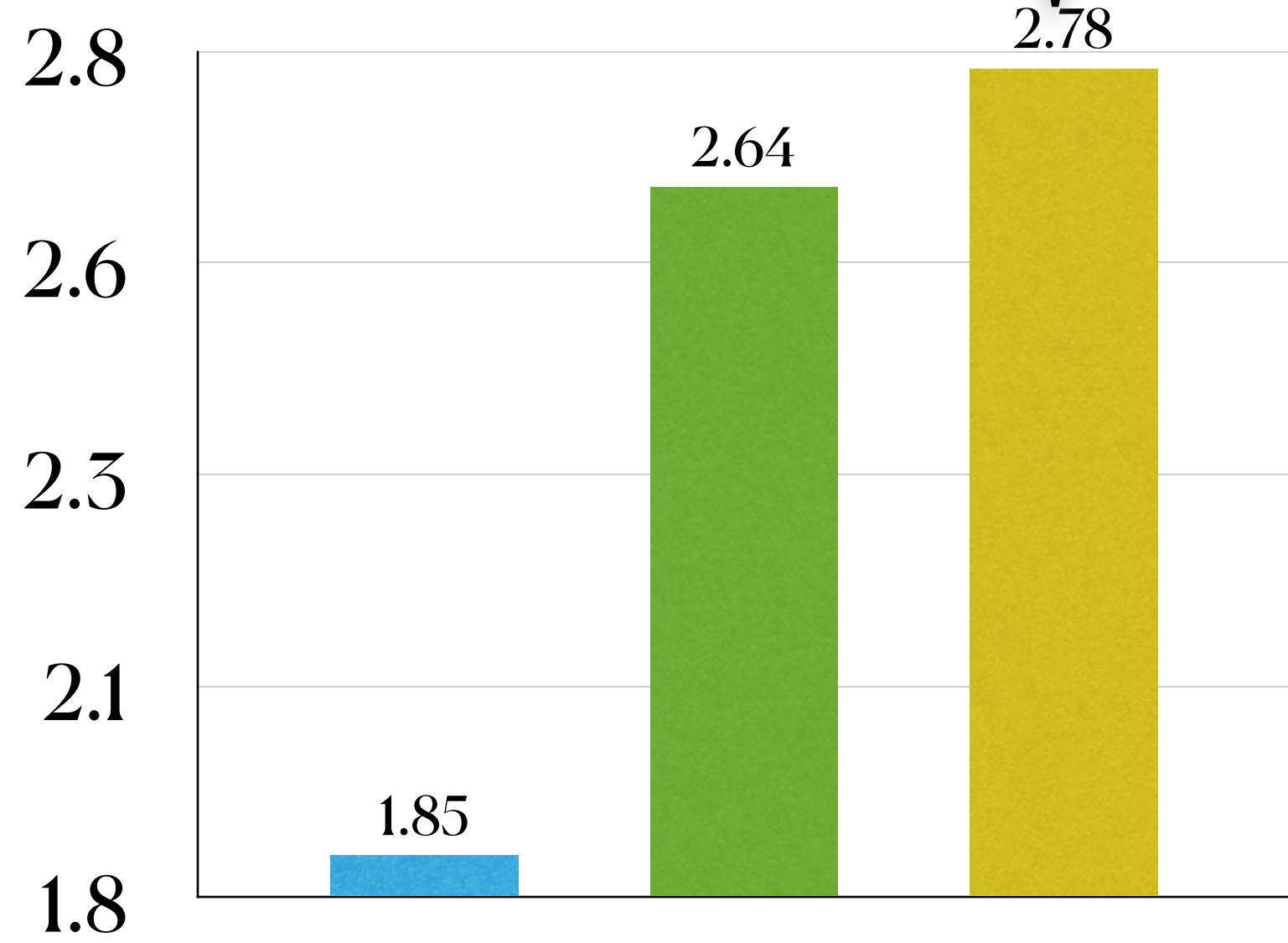


Quality

Off-the-shelf GPT-2

- TSMH
- NeuroLogic
- NeuroLogic A*esq (greedy)

Off-the-shelf A* outperforms all fine-tuned methods



Quality

Enables many constrained generation tasks

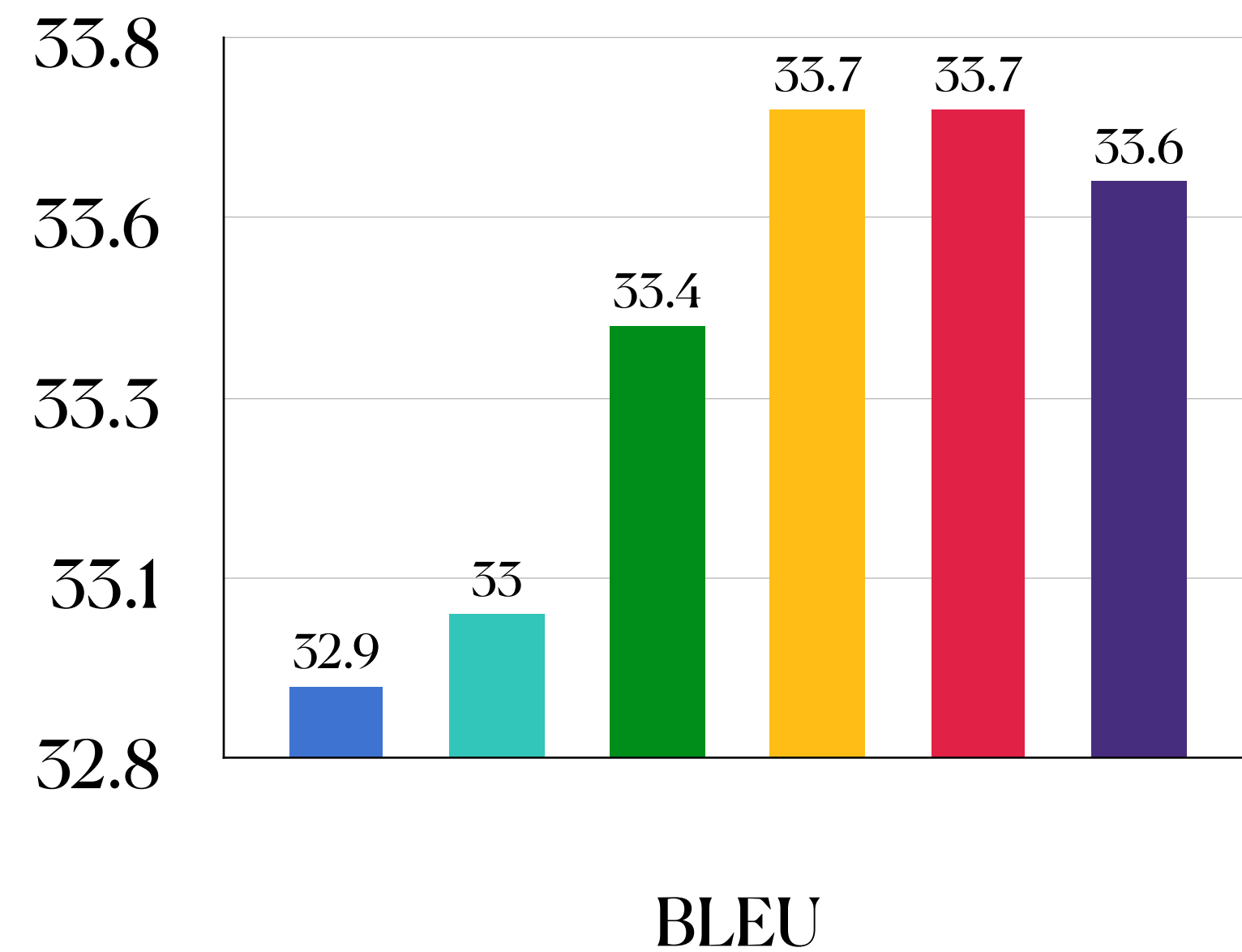


Enables many constrained generation tasks

Constrained MT

(Dinu et al., 2019)

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

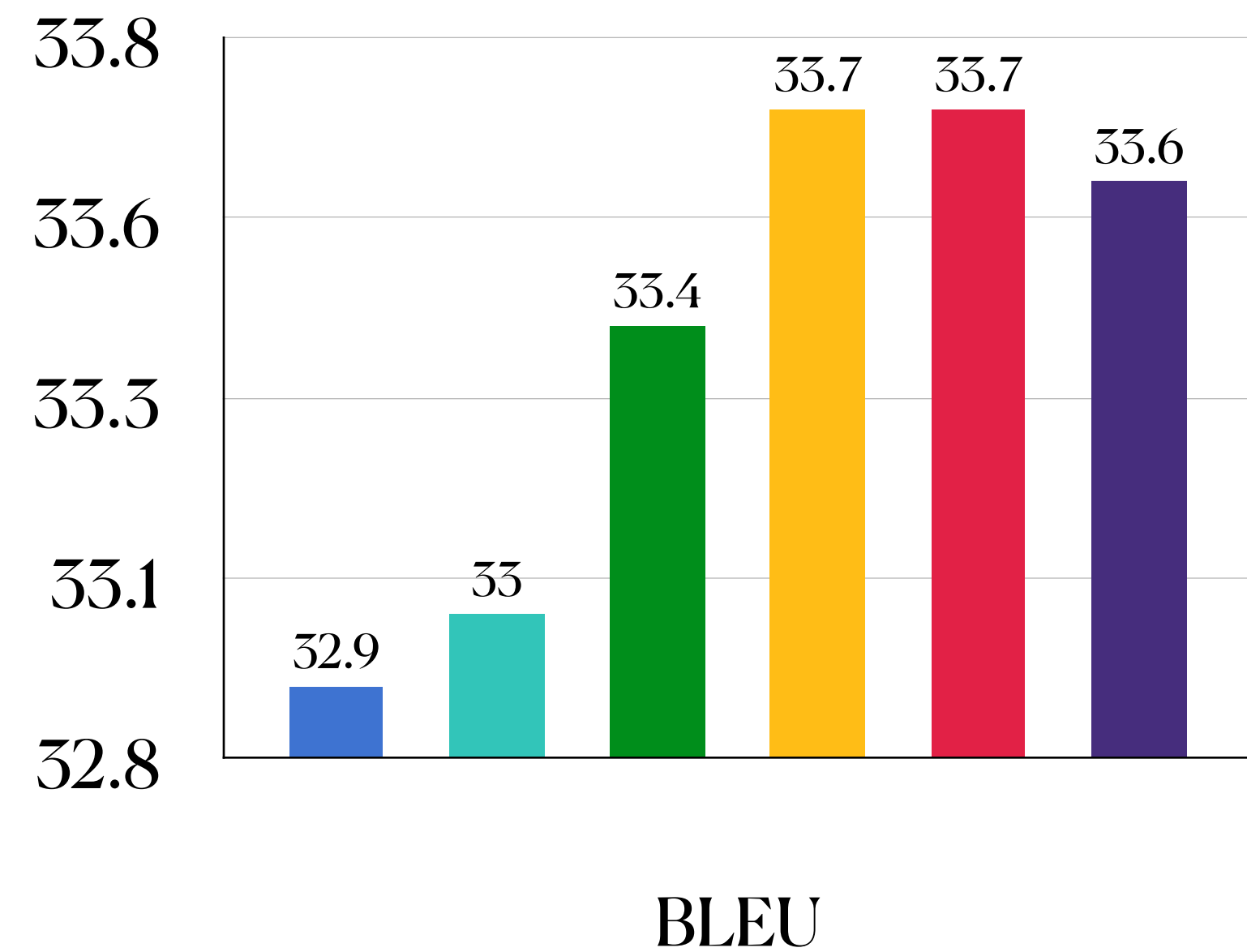


Enables many constrained generation tasks

Constrained MT

(Dinu et al., 2019)

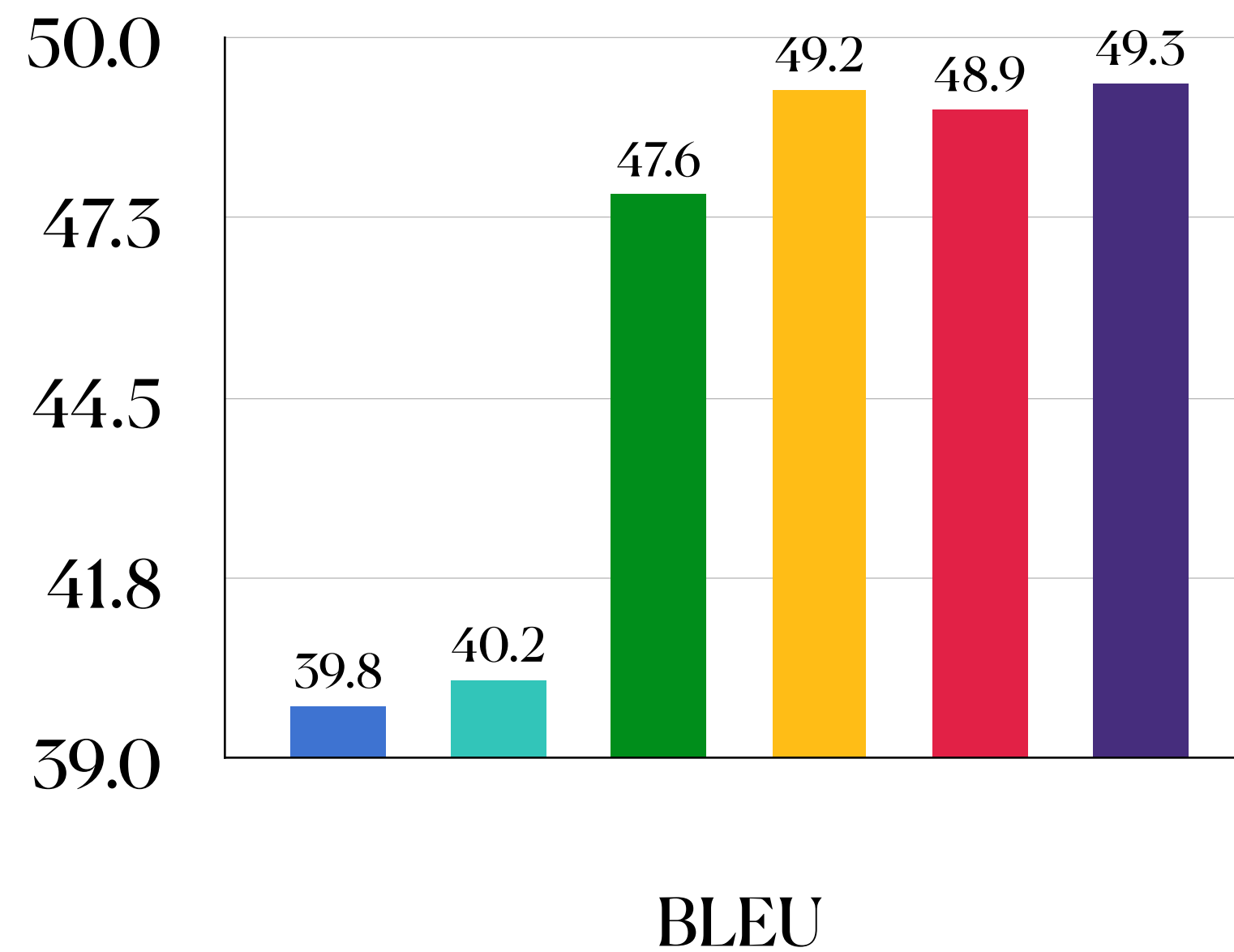
- MarianMT
- Post and Vilar (2018)
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)



Few-Shot E2ENLG

(Chen et al., 2020)

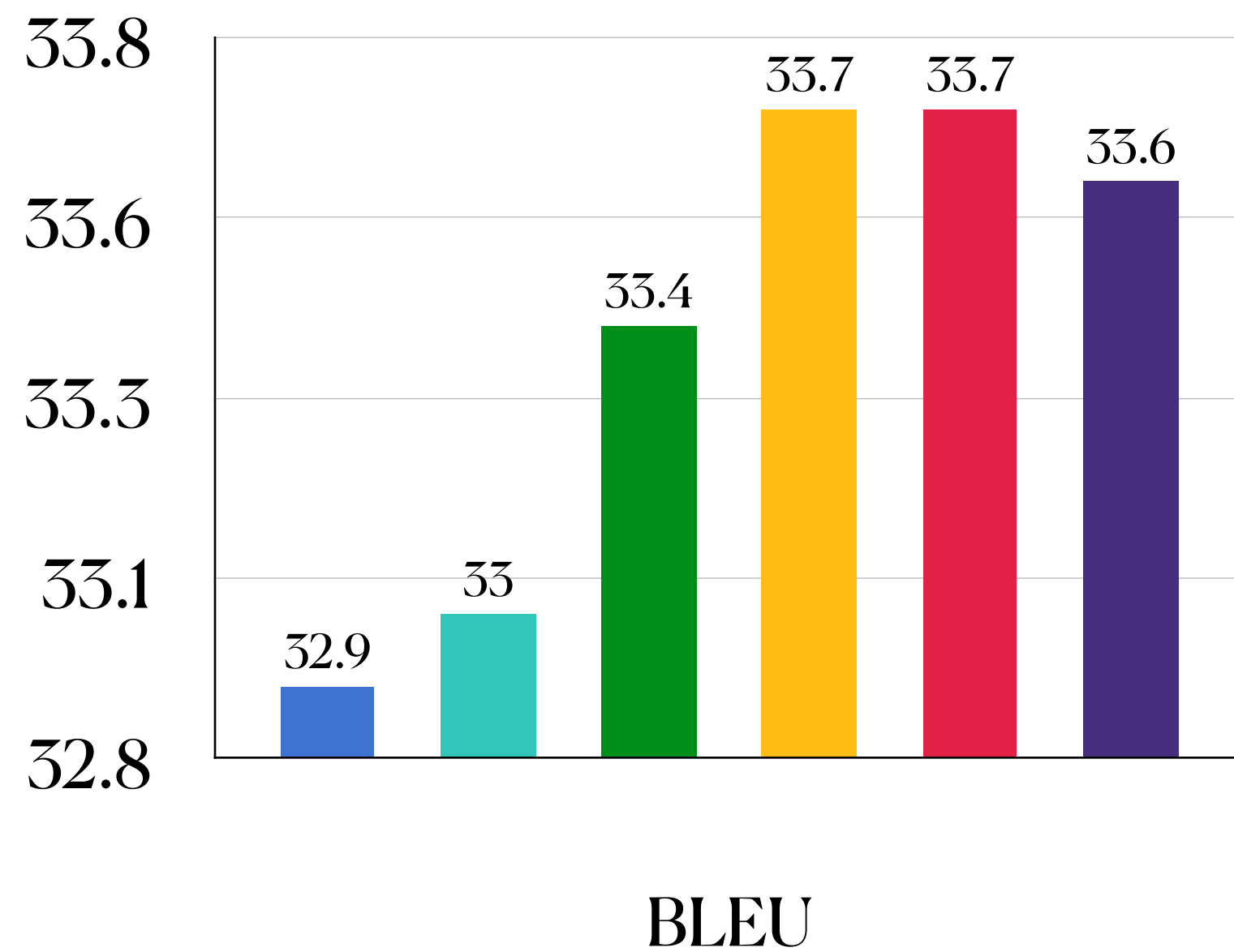
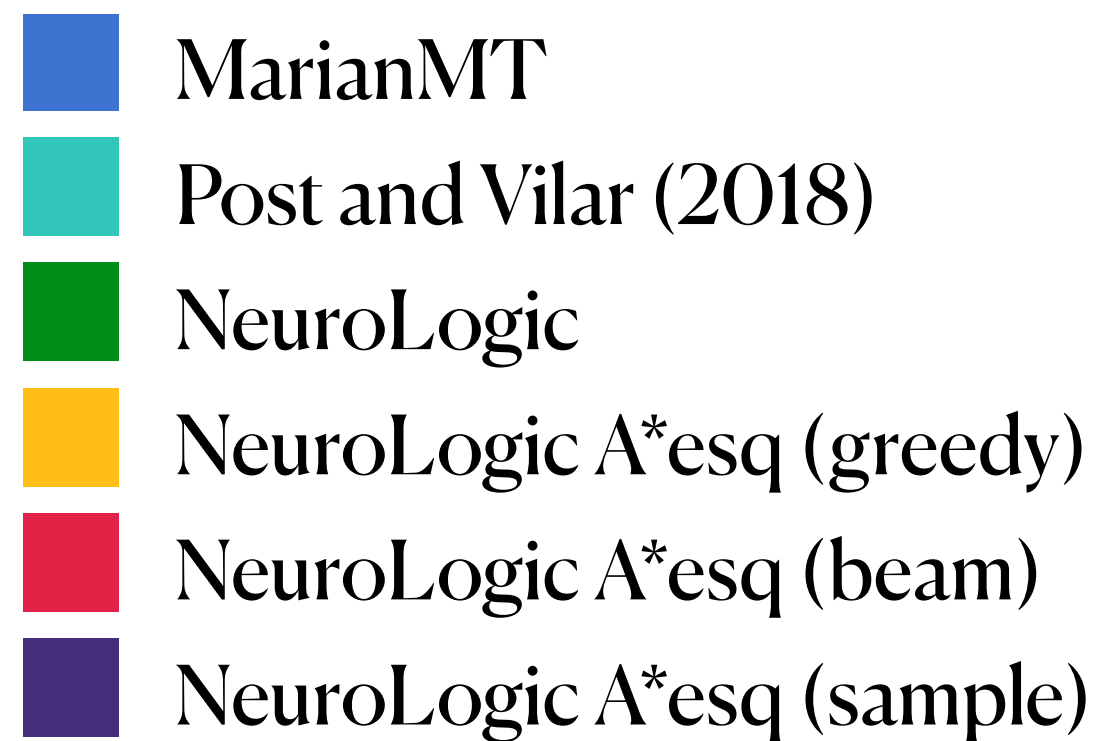
- KGPT-Graph
- KGPT-Seq
- NeuroLogic
- NeuroLogic A*esq (greedy)
- NeuroLogic A*esq (beam)
- NeuroLogic A*esq (sample)



Enables many constrained generation tasks

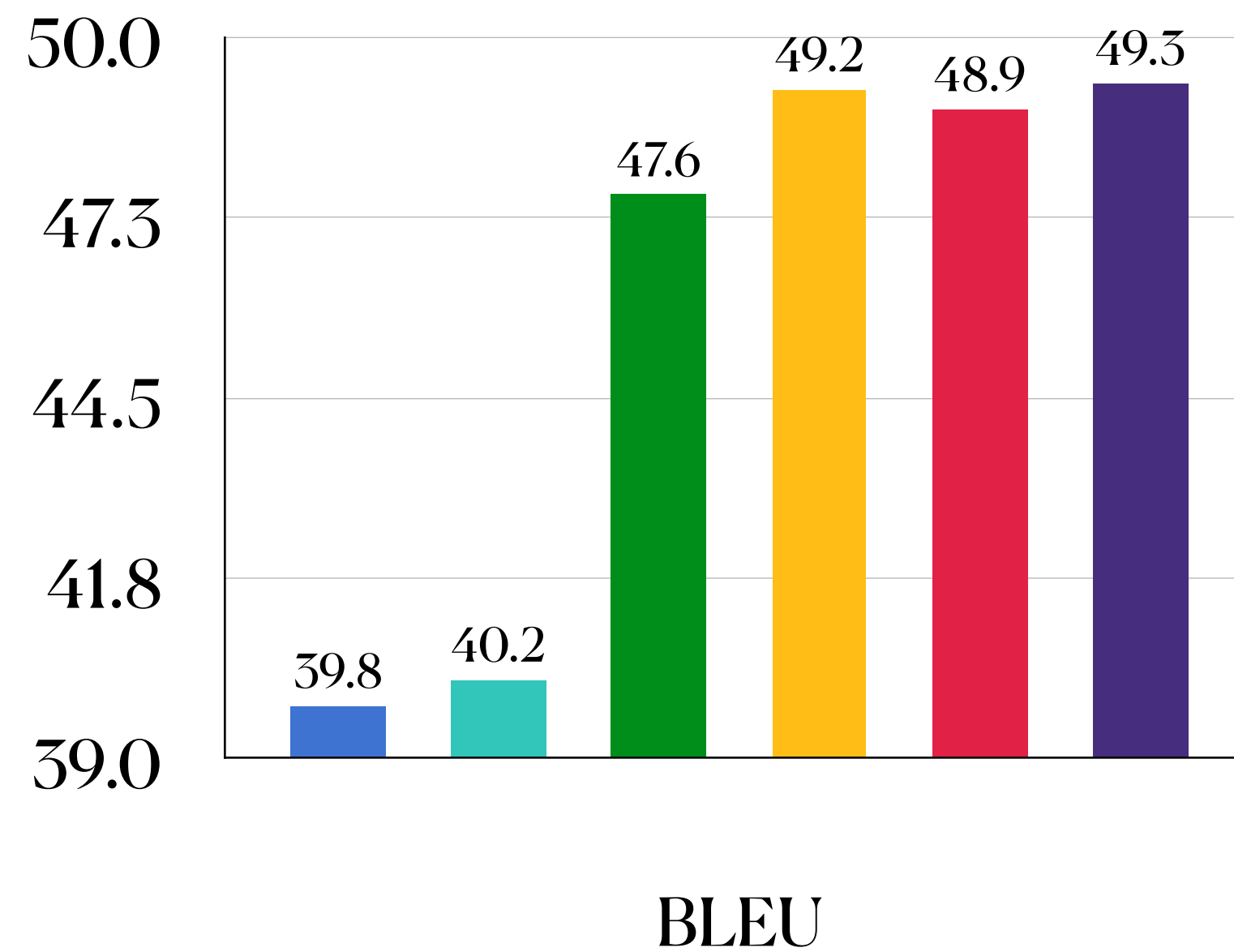
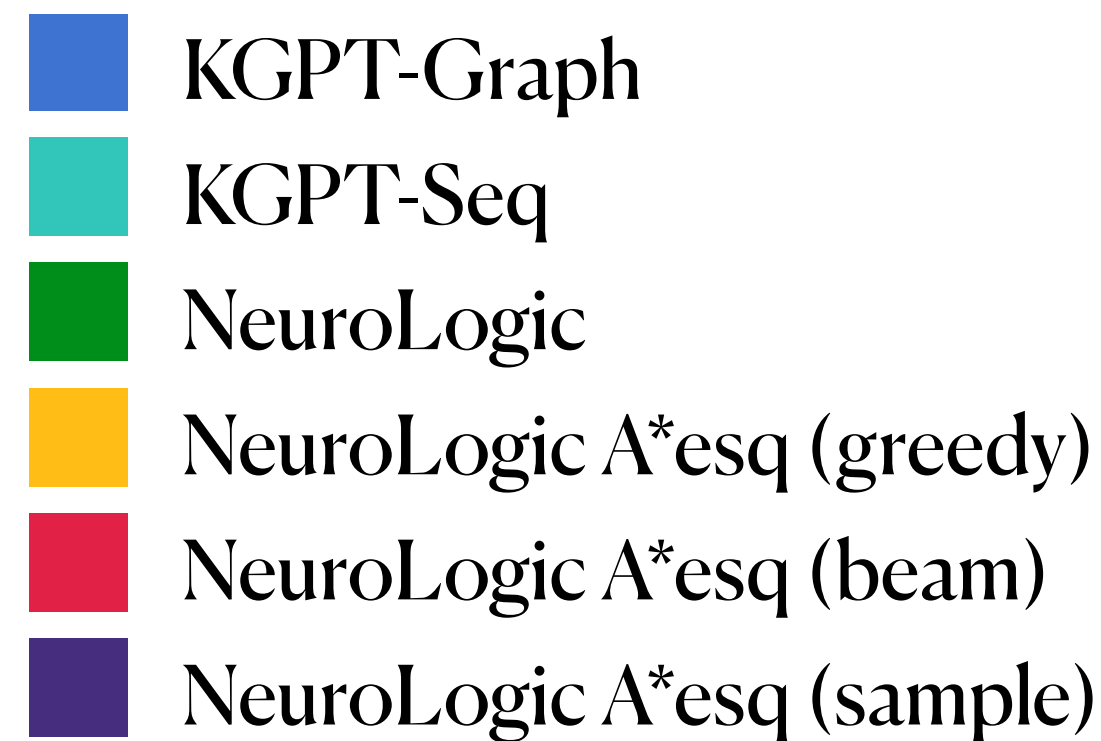
Constrained MT

(Dinu et al., 2019)



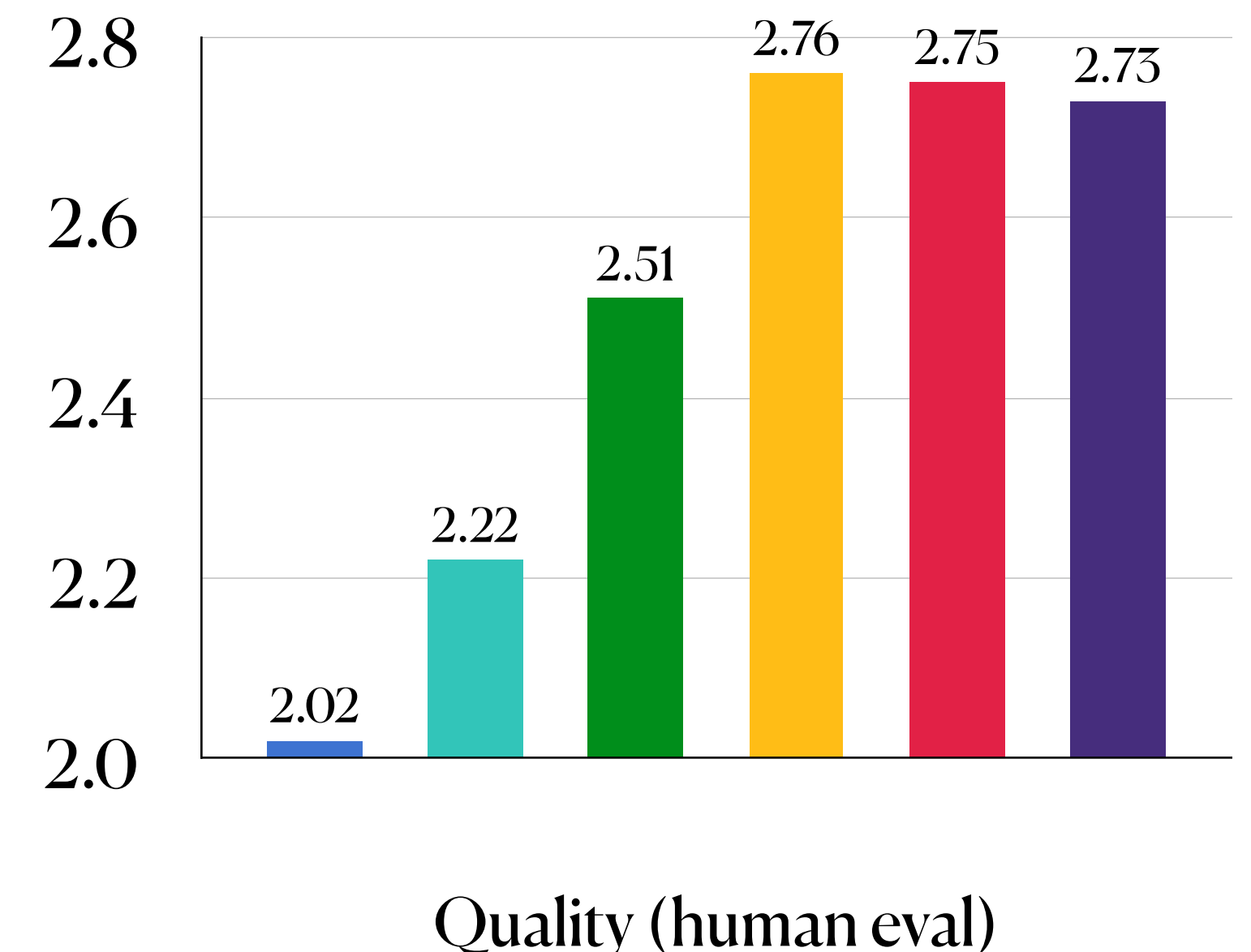
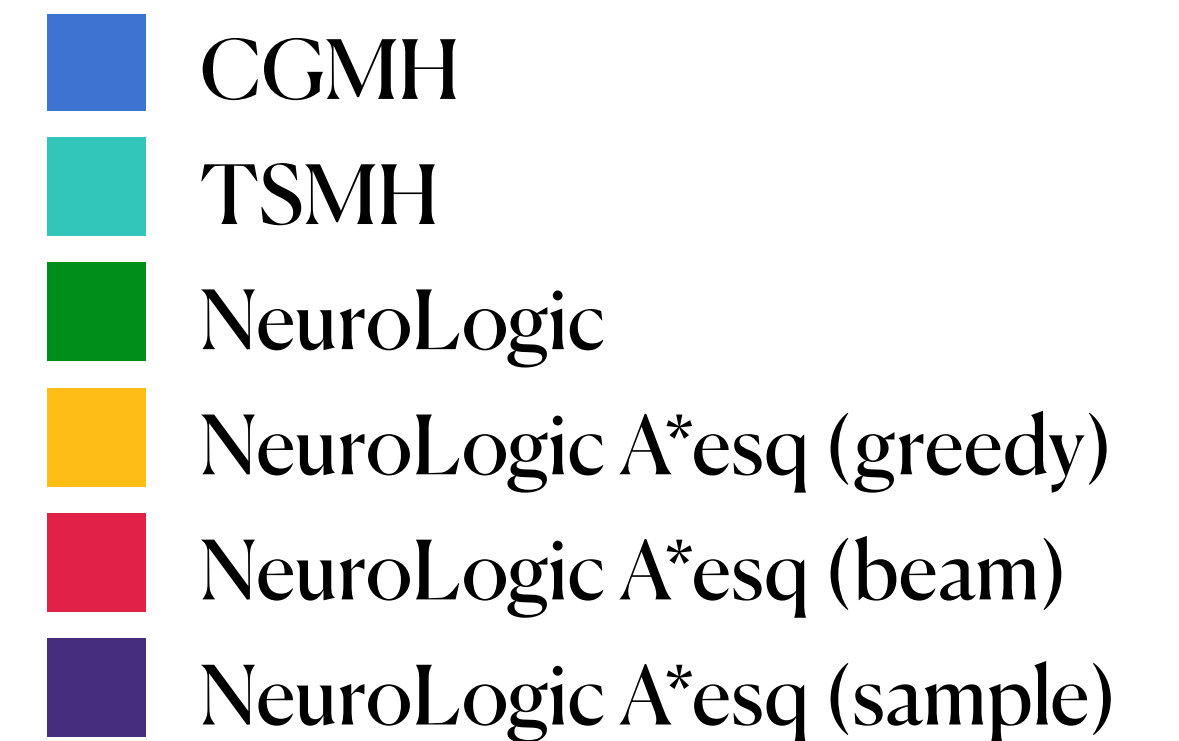
Few-Shot E2ENLG

(Chen et al., 2020)

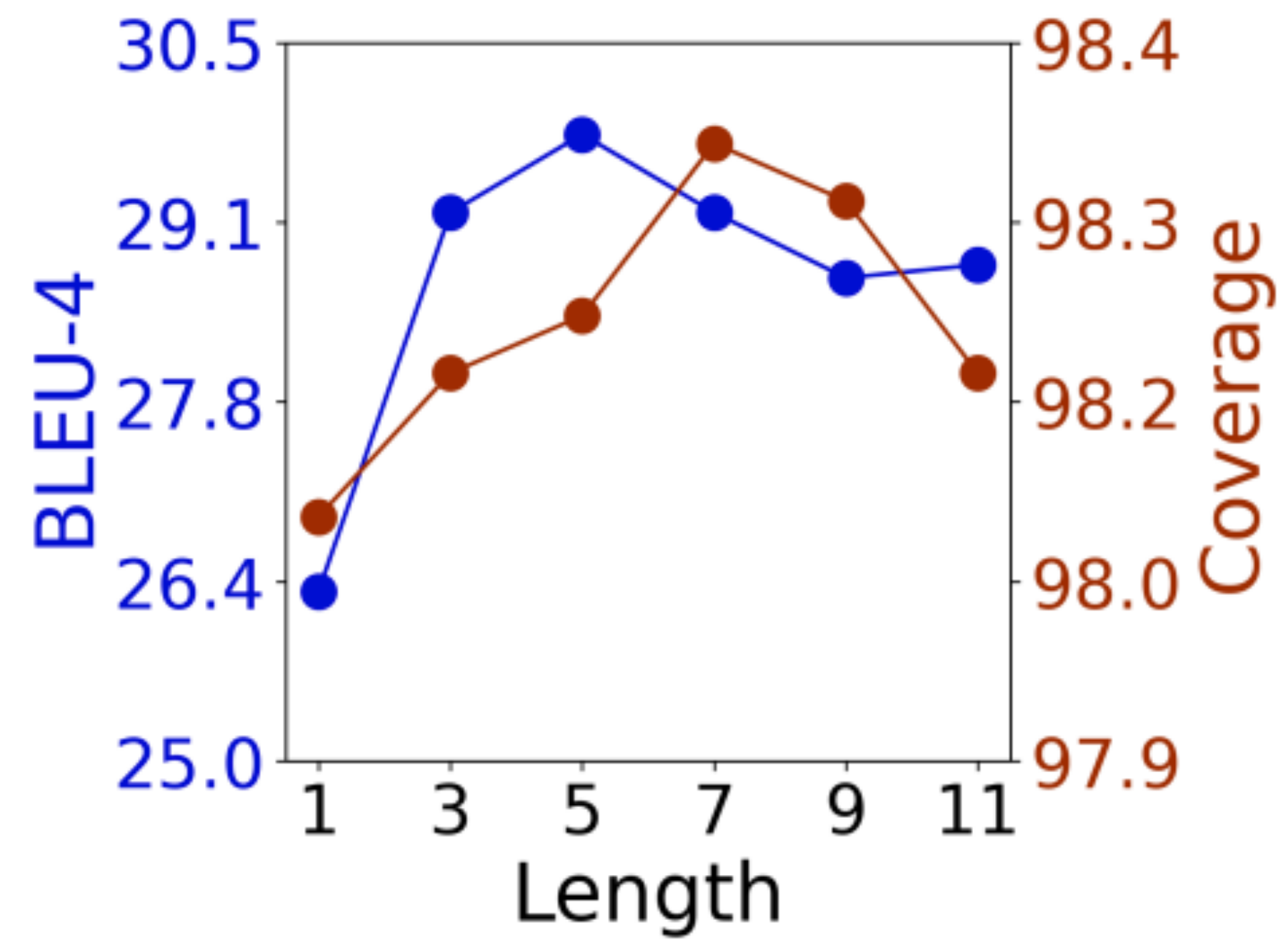


Question Generation

(Zhang et al., 2020)



- Greedy lookahead length (CommonGen)



- Improves at varying amounts of training data

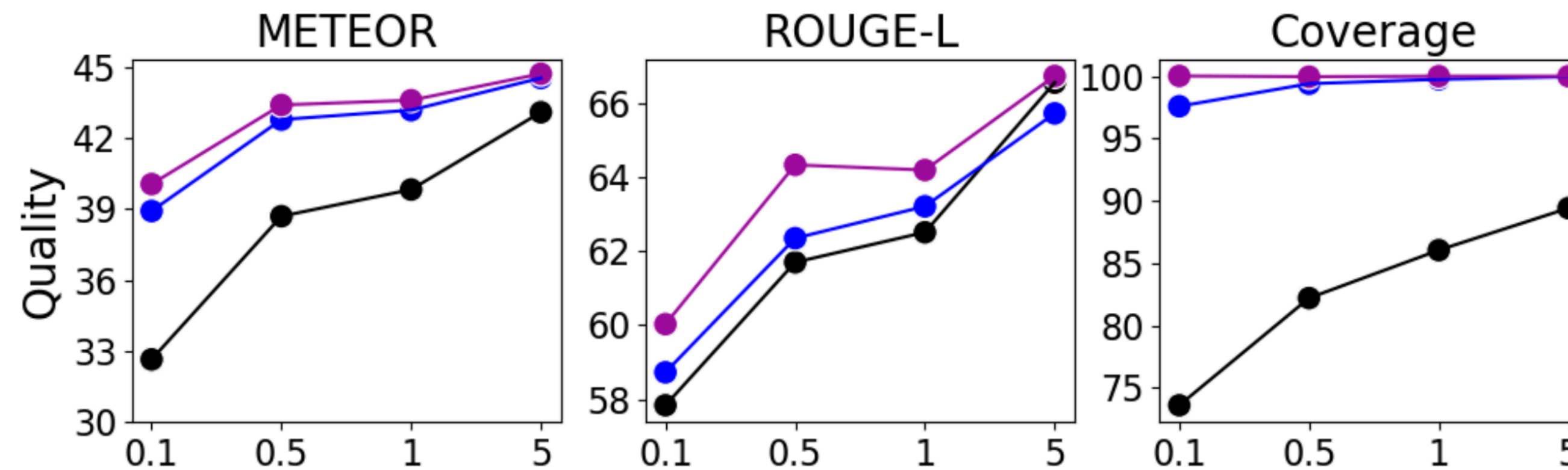
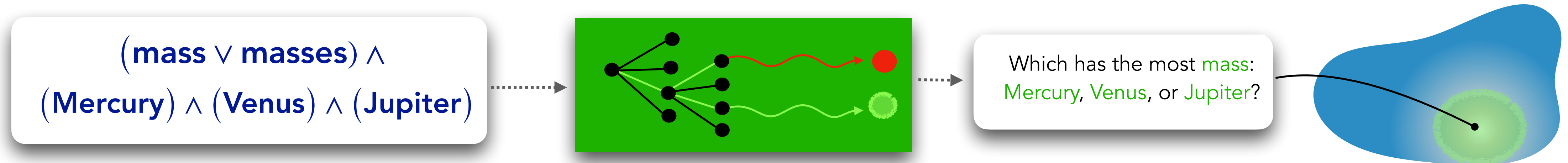


Figure 3: Performance (y-axis) of supervised GPT-2 on E2ENLG, with a varying amount of training data for supervision (x-axis). The **purple**, **blue**, and **black** line denote decoding with NEUROLOGIC[★], NEUROLOGIC and conventional beam search respectively.

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



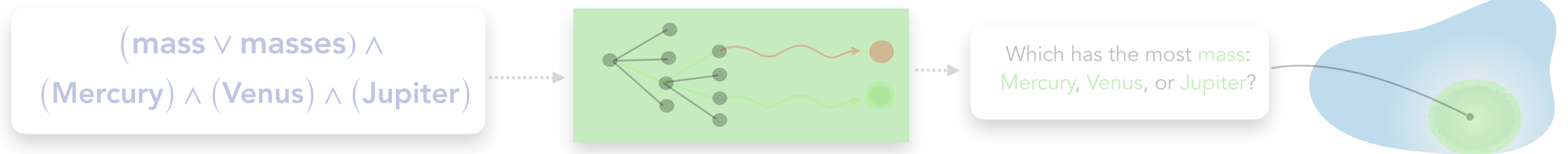
**NeuroLogic A*esque Decoding:
Constrained Text Generation with Lookahead Heuristics**

[arxiv:2112.08726](https://arxiv.org/abs/2112.08726)

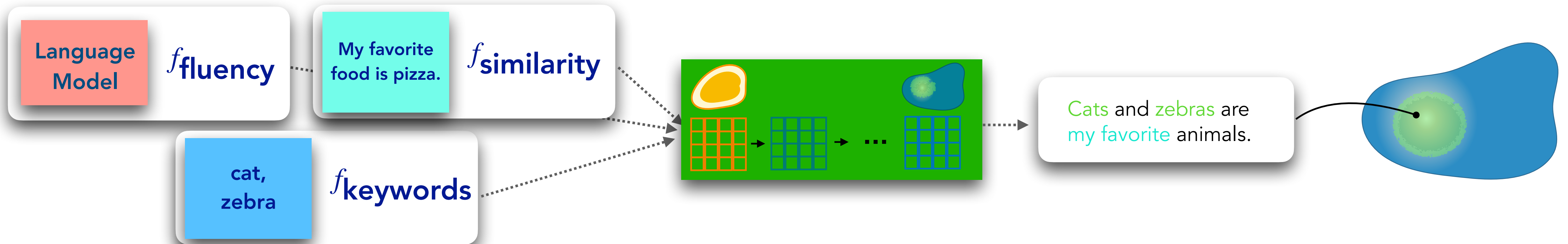
github.com/GloriaXimingLu/star_neurologic

Constrained generation through inference

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



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



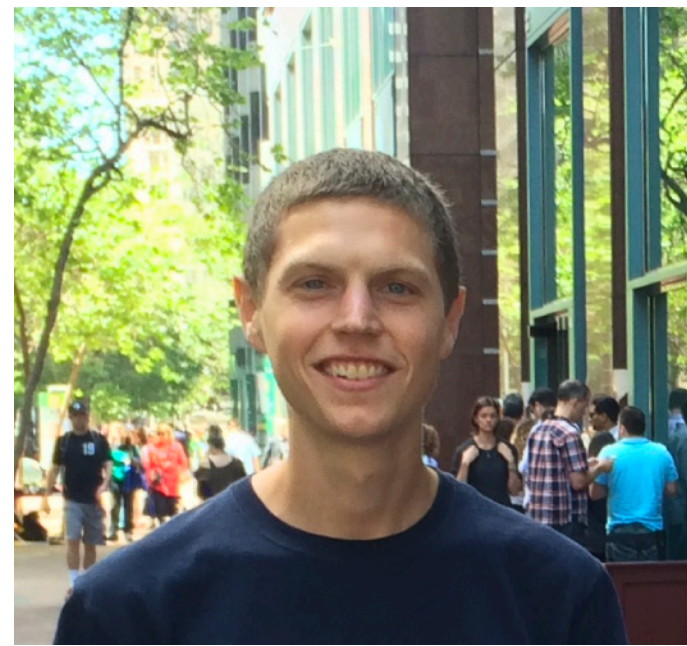
Constrained generation through *continuous* inference

COLD Decoding:
Constrained Decoding with Langevin Dynamics

In Submission, [arxiv:2202.11705](https://arxiv.org/abs/2202.11705)



Lianhui Qin



Sean Welleck



Daniel Khashabi



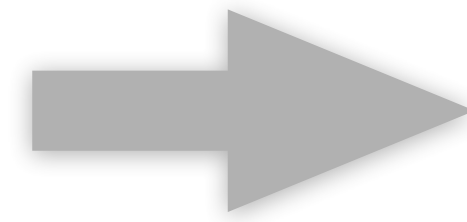
Yejin Choi



Lexically Constrained Generation

Keywords

{ mass, Mercury, Jupiter }



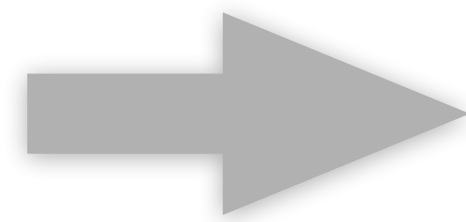
Generation

Jupiter has more mass than
Mercury.

Lexically Constrained Generation

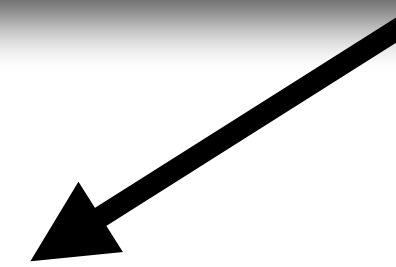
Keywords

{ mass, Mercury, Jupiter }



Generation

Jupiter has more mass than
Mercury.



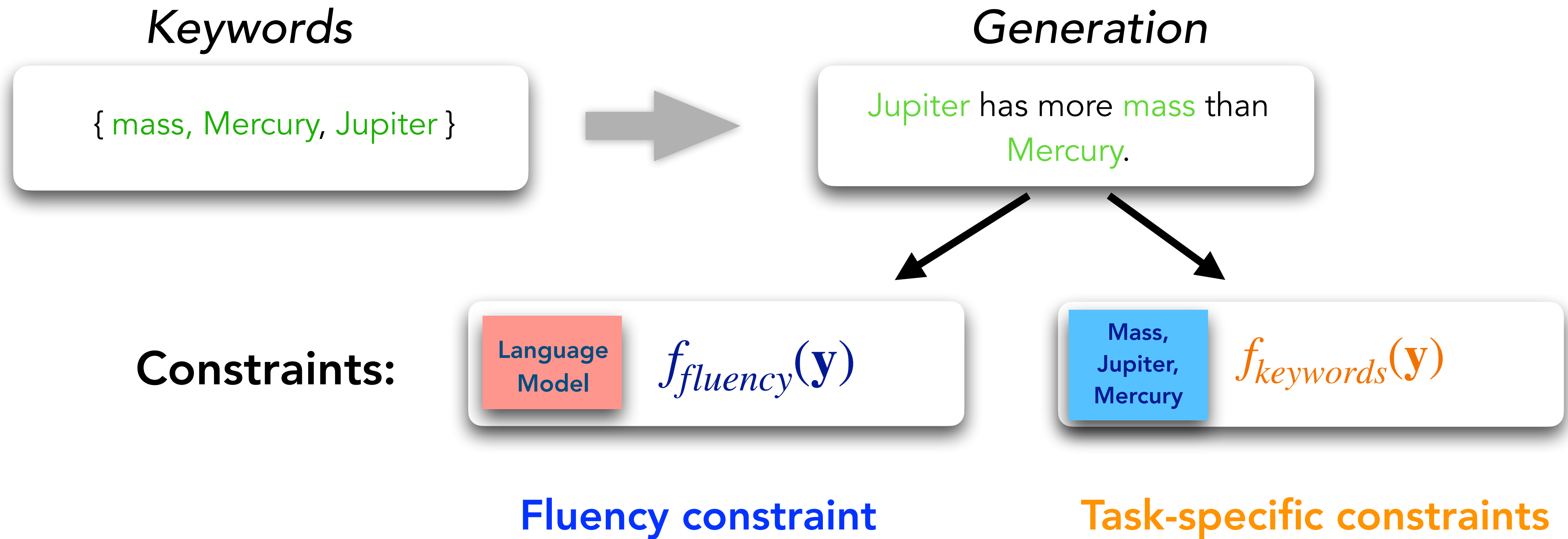
Constraints:

Language
Model

$f_{fluency}(\mathbf{y})$

Fluency constraint

Lexically Constrained Generation



Text infilling / abductive reasoning

AbductiveNLG

(Bhagavatula et al., 2020)

Left context

She went to practice everyday.

Text infilling / abductive reasoning

AbductiveNLG

(Bhagavatula et al., 2020)

Left context

She went to practice everyday.

Right context

She won a gold medal in the
Olympic marathon.

Text infilling / abductive reasoning

AbductiveNLG

(Bhagavatula et al., 2020)

Left context

She went to practice everyday.

Right context

She won a gold medal in the Olympic marathon.

Generation

She ran a lot of miles at practice.

```
graph TD; L[Left context: She went to practice everyday.] --> G[Generation: She ran a lot of miles at practice.]; R[Right context: She won a gold medal in the Olympic marathon.] --> G;
```

Text infilling / abductive reasoning

AbductiveNLG

(Bhagavatula et al., 2020)

Left context

She went to practice everyday.

Right context

She won a gold medal in the Olympic marathon.

Generation

She ran a lot of miles at practice.

Constraints:

Language
Model

$f_{fluency}(y)$

Fluency constraint

Text infilling / abductive reasoning

AbductiveNLG

(Bhagavatula et al., 2020)

Left context

She went to practice everyday.

Right context

She won a gold medal in the Olympic marathon.

Generation

She ran a lot of miles at practice.

Constraints:

Language Model

$f_{\text{fluency}}(\mathbf{y})$

She went to practice ...

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

Fluency constraint

Task-specific constraints

Text infilling / abductive reasoning

AbductiveNLG

(Bhagavatula et al., 2020)

Left context

She went to practice everyday.

Right context

She won a gold medal in the Olympic marathon.

Generation

She ran a lot of miles at practice.

Constraints:

Language Model

$f_{\text{fluency}}(\mathbf{y})$

Fluency constraint

She went to practice ...

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

She won a gold ...

$f_{\text{coherence-right}}(\mathbf{y})$

Task-specific constraints

Text similarity / counterfactual reasoning

TimeTravel

(Qin et al., 2019)

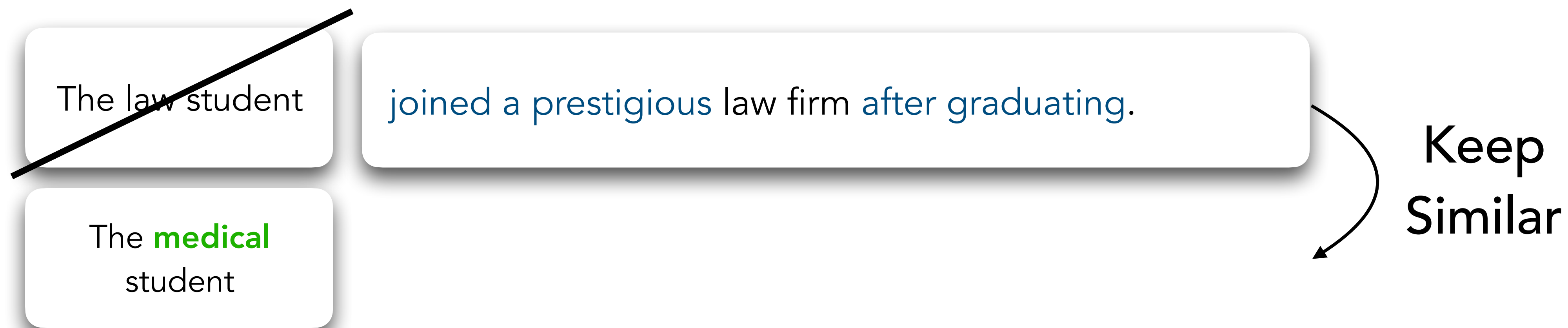
The law student

joined a prestigious law firm after graduating.

Text similarity / counterfactual reasoning

TimeTravel

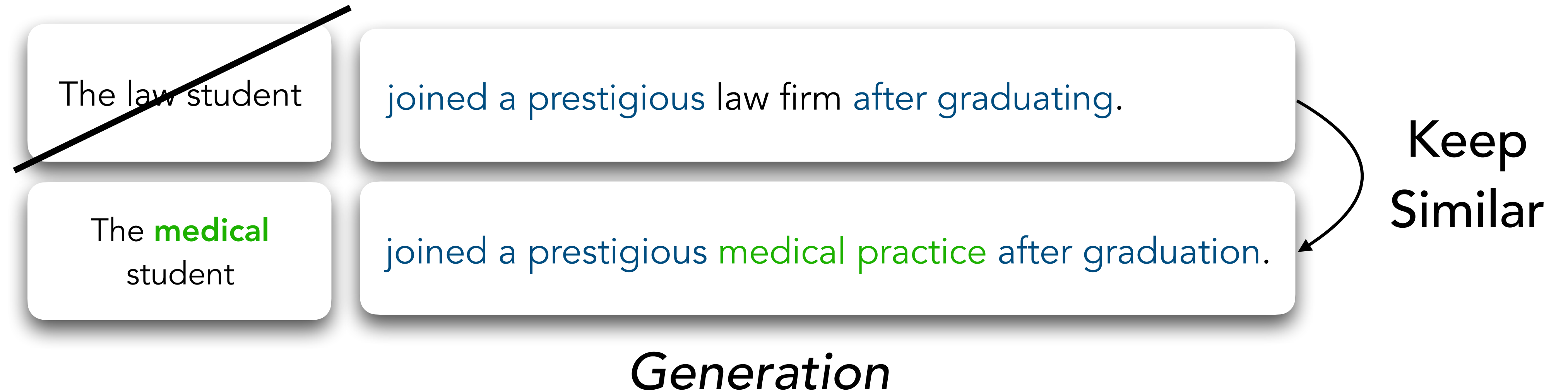
(Qin et al., 2019)



Text similarity / counterfactual reasoning

TimeTravel

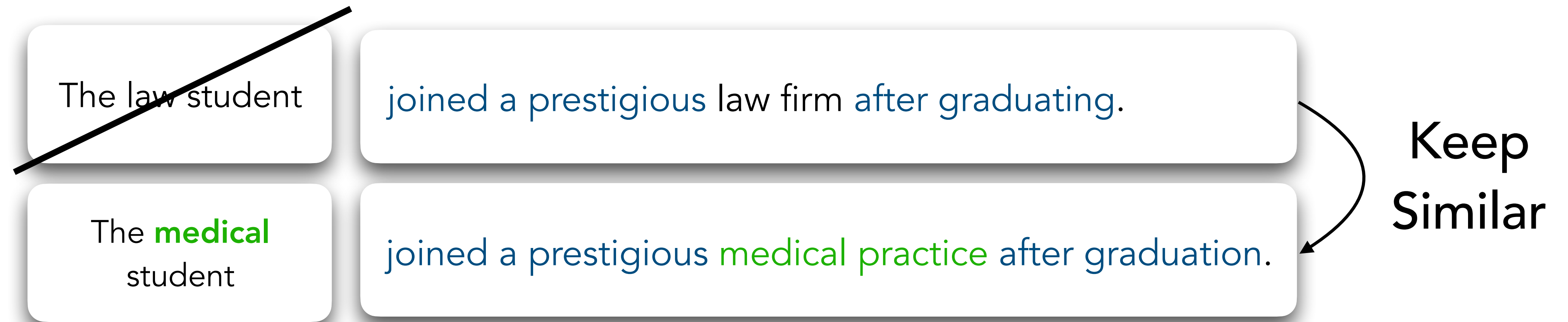
(Qin et al., 2019)



Text similarity / counterfactual reasoning

TimeTravel

(Qin et al., 2019)



Generation

Constraints:

Language
Model

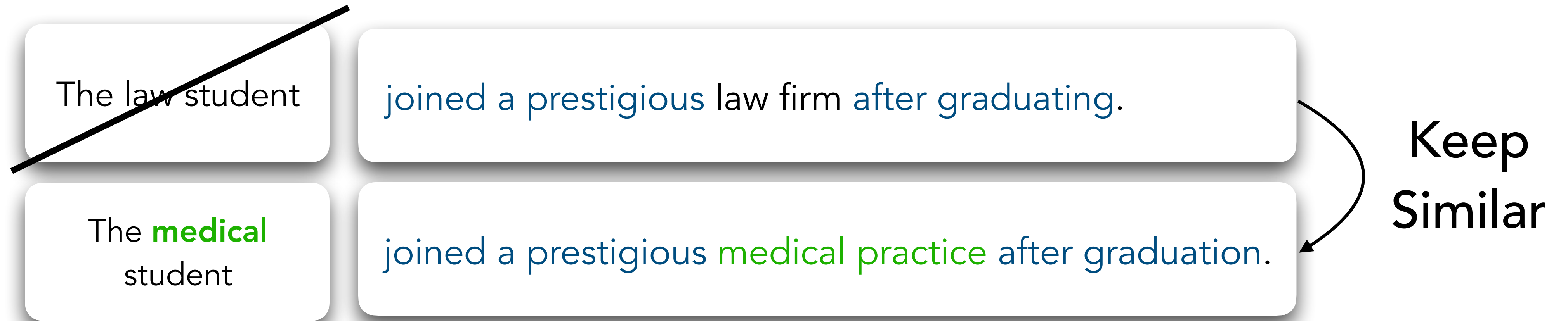
$f_{fluency}(y)$

Fluency constraint

Text similarity / counterfactual reasoning

TimeTravel

(Qin et al., 2019)



Generation

Constraints:

Language
Model

$f_{fluency}(y)$

The medical
student

$f_{coherence-left}(y)$

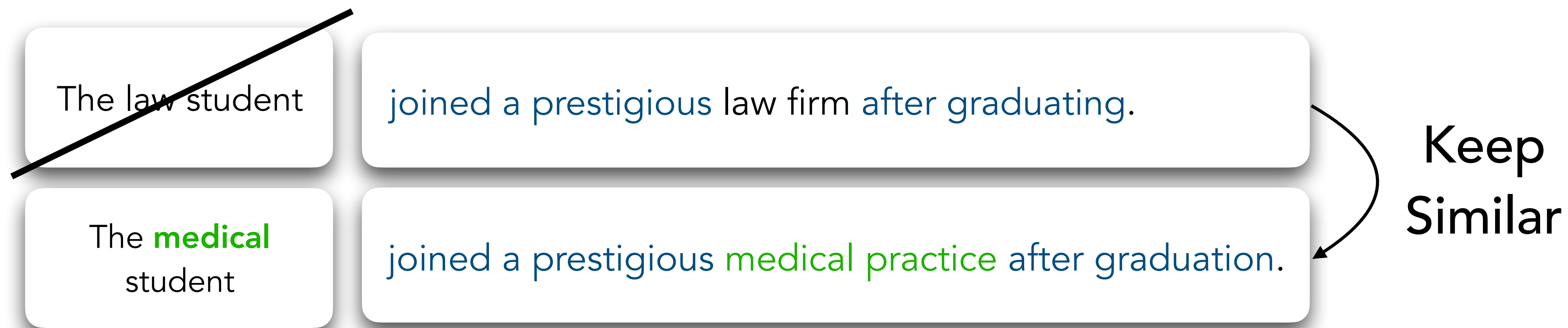
Fluency constraint

Task-specific constraints

Text similarity / counterfactual reasoning

TimeTravel

(Qin et al., 2019)



Generation

Constraints:

Language Model $f_{fluency}(y)$

Fluency constraint

The medical student $f_{coherence-left}(y)$

Joined a prestigious ... $f_{similarity}(y, y^*)$

Task-specific constraints

Constrained generation as sampling from an energy-based model

Fluency constraint

Task-specific constraints

Energy function:

$$E(\mathbf{y}) = f_{fluency}(\mathbf{y}) + f_1(\mathbf{y}) + f_2(\mathbf{y}) + \dots$$

Constrained generation as sampling from an energy-based model

Fluency constraint

Task-specific constraints

Energy function:

$$E(\mathbf{y}) = f_{fluency}(\mathbf{y}) + f_1(\mathbf{y}) + f_2(\mathbf{y}) + \dots$$

Energy-based model:

$$p(\mathbf{y}) = \exp \{ -E(\mathbf{y}) \} / Z$$

Constrained generation as sampling from an energy-based model

Fluency constraint

Task-specific constraints

Energy function:

$$E(\mathbf{y}) = f_{fluency}(\mathbf{y}) + f_1(\mathbf{y}) + f_2(\mathbf{y}) + \dots$$

Energy-based model:

$$p(\mathbf{y}) = \exp \{ -E(\mathbf{y}) \} / Z$$

Constrained generation: $\hat{\mathbf{y}} \sim p(\mathbf{y})$

Sampling from an energy-based model

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

- Gradient-free MCMC (e.g. Gibbs sampling [Bishop & Nasrabadi 2006]): **slow**

Sampling from an energy-based model

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

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

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

Sampling from an energy-based model

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

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

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

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

Sampling from an energy-based model

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

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

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

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

$\nabla_{\mathbf{y}} E(\mathbf{y})$ not defined for discrete \mathbf{y}

Sampling from an energy-based model

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

- Define energy over “**soft sequence**” of continuous vectors:
 - $\tilde{\mathbf{y}} = (\tilde{\mathbf{y}}_1, \dots, \tilde{\mathbf{y}}_T)$, where $\tilde{\mathbf{y}}_t \in \mathbf{R}^{vocab}$

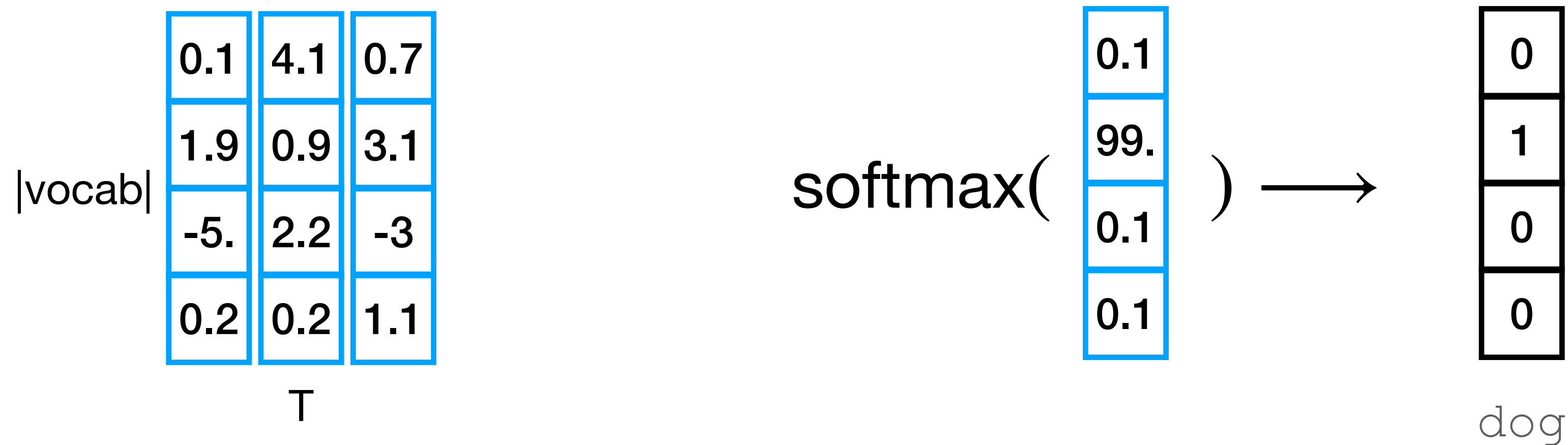
	0.1	4.1	0.7
	1.9	0.9	3.1
vocab	-5.	2.2	-3
	0.2	0.2	1.1
	T		

Sampling from an energy-based model

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

- Define energy over “**soft sequence**” of continuous vectors:

- $\tilde{\mathbf{y}} = (\tilde{\mathbf{y}}_1, \dots, \tilde{\mathbf{y}}_T)$, where $\tilde{\mathbf{y}}_t \in \mathbf{R}^{vocab}$



- Discrete token: $\text{softmax}(\tilde{\mathbf{y}}_t / \tau)$ as $\tau \rightarrow 0$

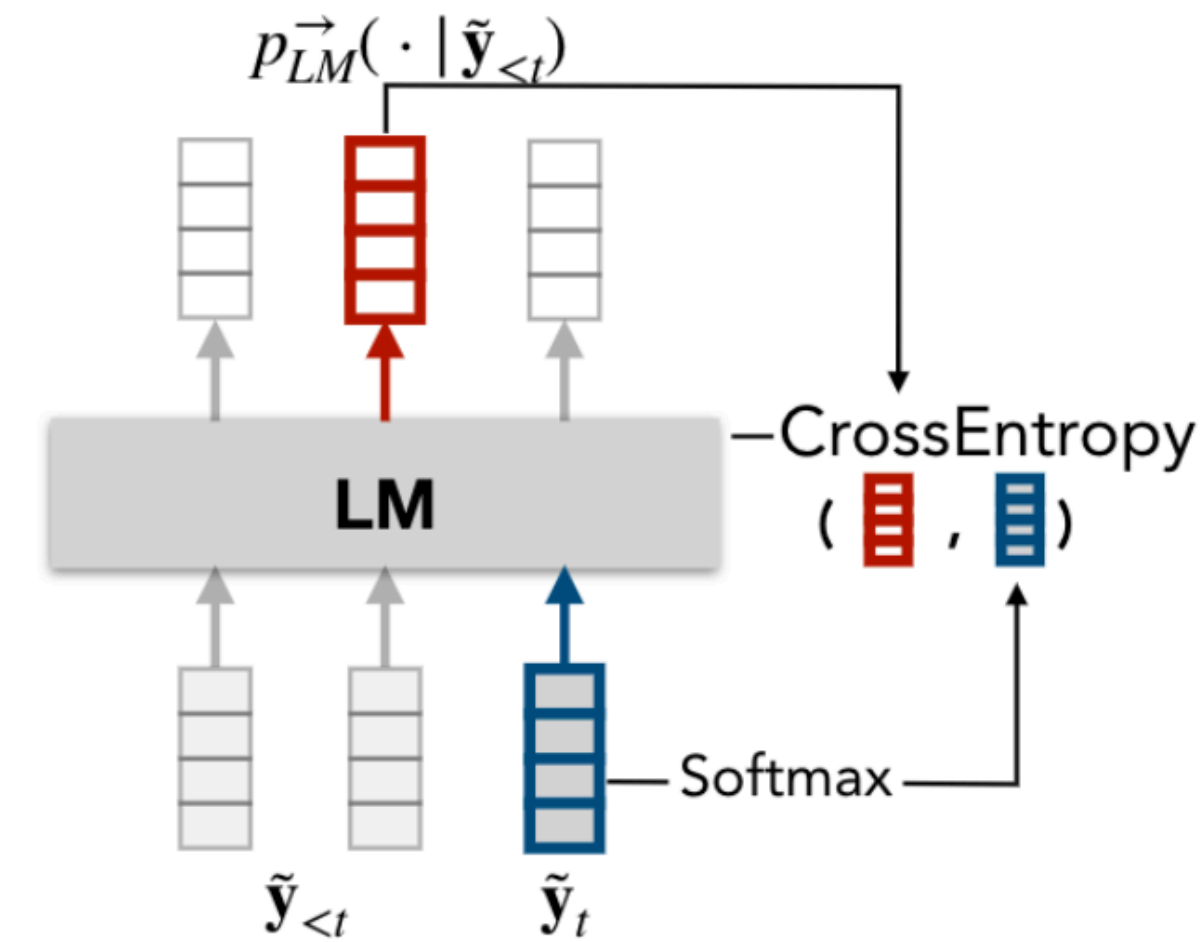
Sampling from an energy-based model

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

- Constraints as **differentiable functions**



$$f_{LM}^{\rightarrow}(\tilde{\mathbf{y}}) = \sum_{t=1}^T \sum_{v \in \mathcal{V}} p_{LM}^{\rightarrow}(v | \tilde{\mathbf{y}}_{<t}) \log \text{softmax}(\tilde{\mathbf{y}}_t(v))$$



Sampling from an energy-based model

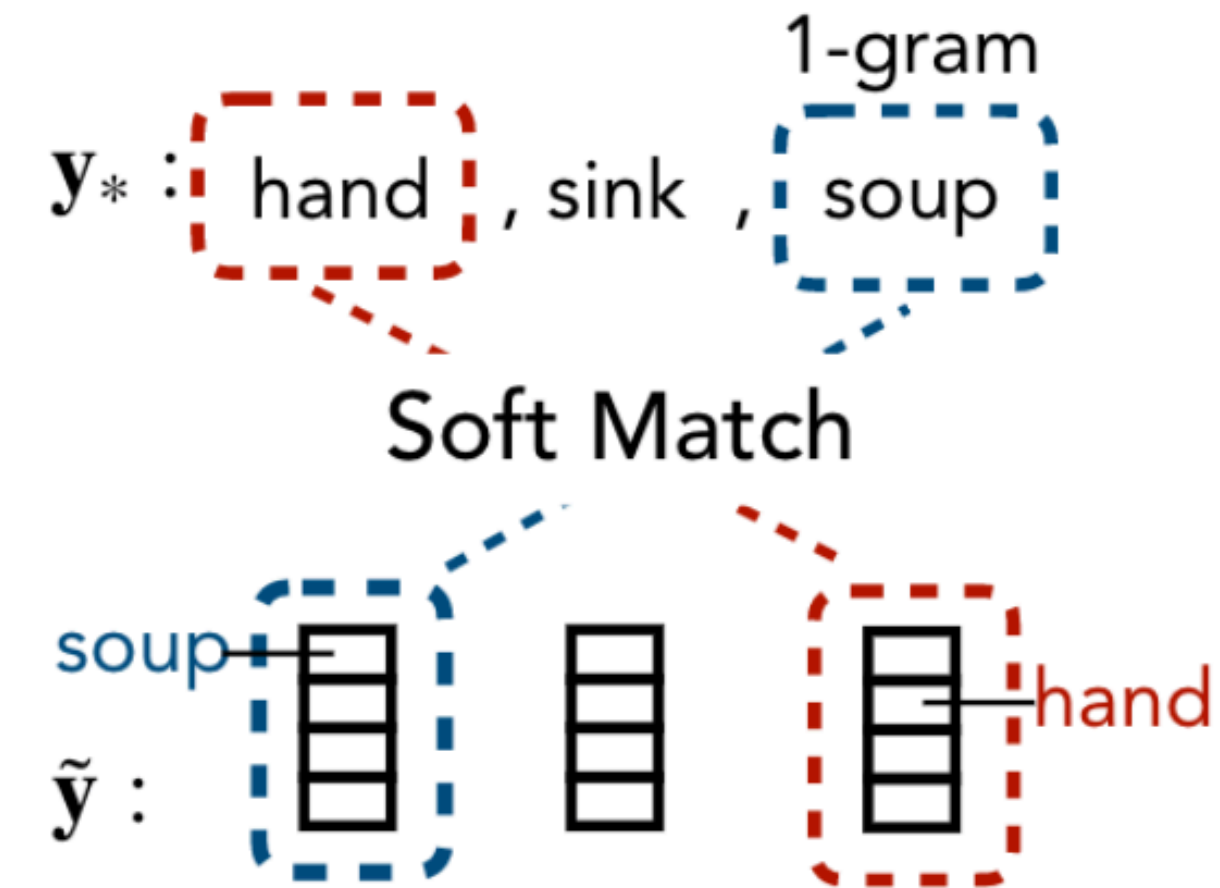
Constrained generation: $\hat{\mathbf{y}} \sim \exp \{ -E(\mathbf{y}) \} / Z$

- Constraints as **differentiable functions**

Mass,
Jupiter,
Mercury

$$f_{keywords}(\tilde{\mathbf{y}})$$

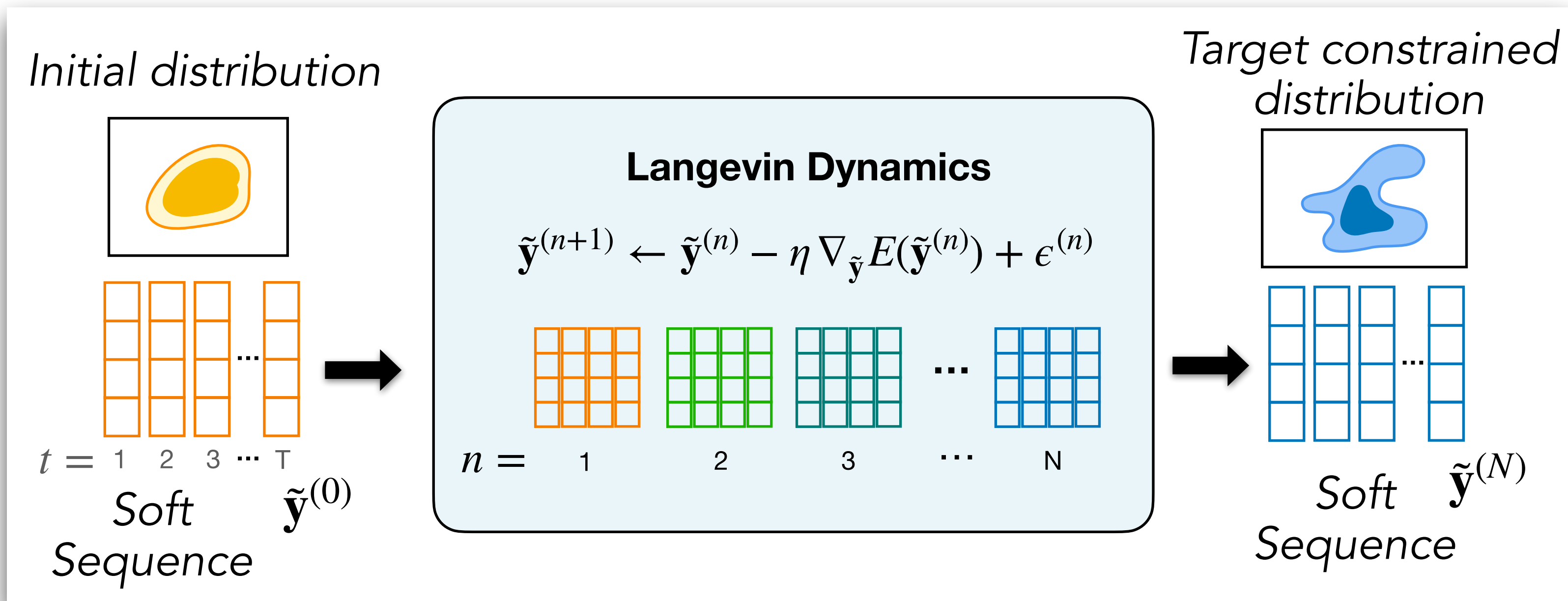
Joined a
prestigious
...

$$f_{similarity}(\tilde{\mathbf{y}}, \mathbf{y}_*)$$


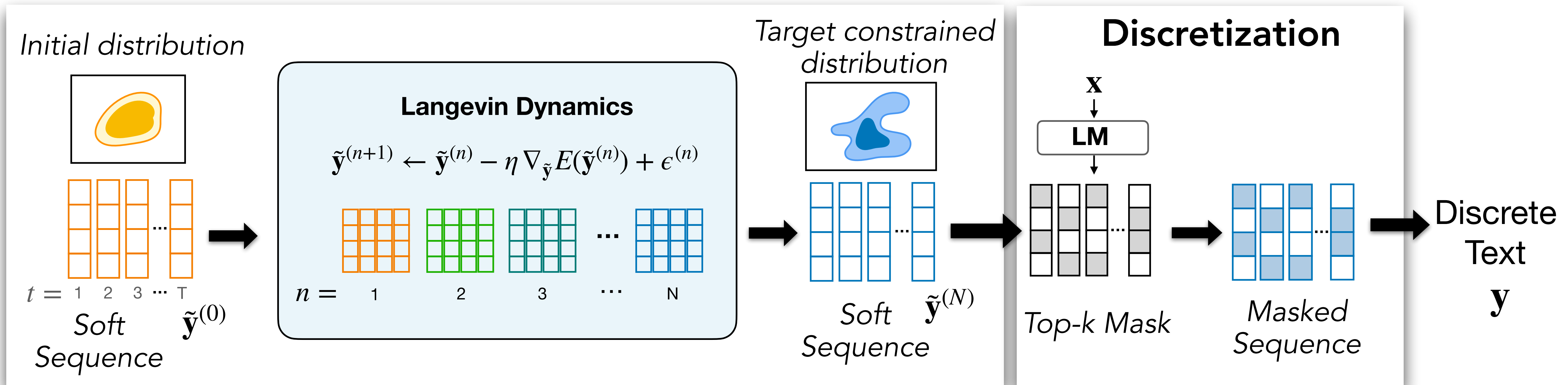
$$f_{sim}(\tilde{\mathbf{y}}; \mathbf{y}_*) = \text{ngram-match}(\tilde{\mathbf{y}}, \mathbf{y}_*), \quad (\text{Liu et al., 2021})$$

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

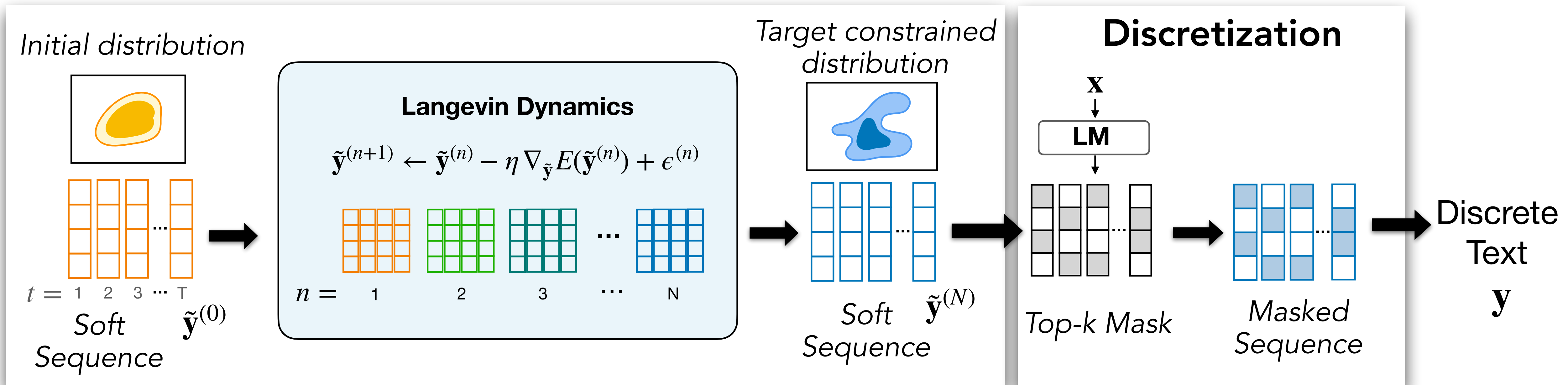
Specify energy $E(\tilde{\mathbf{y}}) = \sum_i f_i(\tilde{\mathbf{y}})$, then:



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



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



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

Lexically constrained generation

CommonGen

(Lin et al., 2020)

We specify an energy function of the following form:

$$E(\tilde{\mathbf{y}}) = \lambda_a^{lr} f_{\text{LM}}^{\rightarrow}(\tilde{\mathbf{y}}) + \lambda_a^{rl} f_{\text{LM}}^{\leftarrow}(\tilde{\mathbf{y}}) + \lambda_b f_{\text{sim}}(\tilde{\mathbf{y}}; \mathcal{W}) + \lambda_c f_{\text{pred}}(\tilde{\mathbf{y}}; c(\mathcal{W})).$$

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

CommonGen

(Lin et al., 2020)

We specify an energy function of the following form:

$$E(\tilde{\mathbf{y}}) = \lambda_a^{lr} f_{\text{LM}}^{\rightarrow}(\tilde{\mathbf{y}}) + \lambda_a^{rl} f_{\text{LM}}^{\leftarrow}(\tilde{\mathbf{y}}) + \lambda_b f_{\text{sim}}(\tilde{\mathbf{y}}; \mathcal{W}) + \lambda_c f_{\text{pred}}(\tilde{\mathbf{y}}; c(\mathcal{W})).$$

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

- Good constraint coverage

Lexically constrained generation

CommonGen

(Lin et al., 2020)

We specify an energy function of the following form:

$$E(\tilde{\mathbf{y}}) = \lambda_a^{lr} f_{\text{LM}}^{\rightarrow}(\tilde{\mathbf{y}}) + \lambda_a^{rl} f_{\text{LM}}^{\leftarrow}(\tilde{\mathbf{y}}) + \lambda_b f_{\text{sim}}(\tilde{\mathbf{y}}; \mathcal{W}) + \lambda_c f_{\text{pred}}(\tilde{\mathbf{y}}; c(\mathcal{W})).$$

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

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

Abductive reasoning

AbductiveNLG

(Bhagavatula et al., 2020)

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

$$E(\tilde{\mathbf{y}}) = \lambda_a^{lr} f_{LM}^{\rightarrow}(\tilde{\mathbf{y}}; \mathbf{x}_l) + \lambda_a^{rl} f_{LM}^{\leftarrow}(\tilde{\mathbf{y}}; \mathbf{x}_r) + \lambda_b f_{\text{pred}}(\tilde{\mathbf{y}}; \mathbf{x}_r) + \lambda_c f_{\text{sim}}(\tilde{\mathbf{y}}; \text{kw}(\mathbf{x}_r) - \text{kw}(\mathbf{x}_l)).$$

Begin. \mathbf{x}_l	Tim wanted to learn astronomy.
End. \mathbf{x}_r	Tim worked hard in school to become one.
LEFT-ONLY	He was a good student.
DELOREAN	So he bought a telescope.
COLD (ours)	He wanted to become a professional astronomer.

Abductive reasoning

AbductiveNLG

(Bhagavatula et al., 2020)

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

$$E(\tilde{\mathbf{y}}) = \lambda_a^{lr} f_{LM}^{\rightarrow}(\tilde{\mathbf{y}}; \mathbf{x}_l) + \lambda_a^{rl} f_{LM}^{\leftarrow}(\tilde{\mathbf{y}}; \mathbf{x}_r) + \lambda_b f_{\text{pred}}(\tilde{\mathbf{y}}; \mathbf{x}_r) + \lambda_c f_{\text{sim}}(\tilde{\mathbf{y}}; \text{kw}(\mathbf{x}_r) - \text{kw}(\mathbf{x}_l)).$$

Begin. \mathbf{x}_l	Tim wanted to learn astronomy.
End. \mathbf{x}_r	Tim worked hard in school to become one.
LEFT-ONLY	He was a good student.
DELOREAN	So he bought a telescope.
COLD (ours)	He wanted to become a professional astronomer.

Models	Automatic Eval				Human Eval			
	BLEU ₄	ROUGE-L	CIDEr	BERTScore	Grammar	Left-coherence ($\mathbf{x}_l\mathbf{y}$)	Right-coherence ($\mathbf{y}\mathbf{x}_r$)	Overall-coherence ($\mathbf{x}_l\mathbf{y}\mathbf{x}_r$)
LEFT-ONLY	0.88	16.26	3.49	38.48	4.57	3.95	2.68	2.70
DELOREAN	1.60	19.06	7.88	41.74	4.30	4.23	2.83	2.87
COLD (ours)	1.79	19.50	10.68	42.67	4.44	4.00	3.06	2.96

Abductive reasoning

AbductiveNLG

(Bhagavatula et al., 2020)

top- k	Grammar	Left-coher. (x-y)	Right-coher. (y-z)	Overall-coher. (x-y-z)
2	4.38	3.99	2.88	2.92
5	4.27	3.71	3.04	2.87
10	4.09	3.84	3.09	2.94
50	3.95	3.62	3.07	2.87
100	3.80	3.54	3.03	2.84

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)

top- k	Grammar	Left-coher. (x-y)	Right-coher. (y-z)	Overall-coher. (x-y-z)
2	4.38	3.99	2.88	2.92
5	4.27	3.71	3.04	2.87
10	4.09	3.84	3.09	2.94
50	3.95	3.62	3.07	2.87
100	3.80	3.54	3.03	2.84

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

AbductiveNLG

(Bhagavatula et al., 2020)

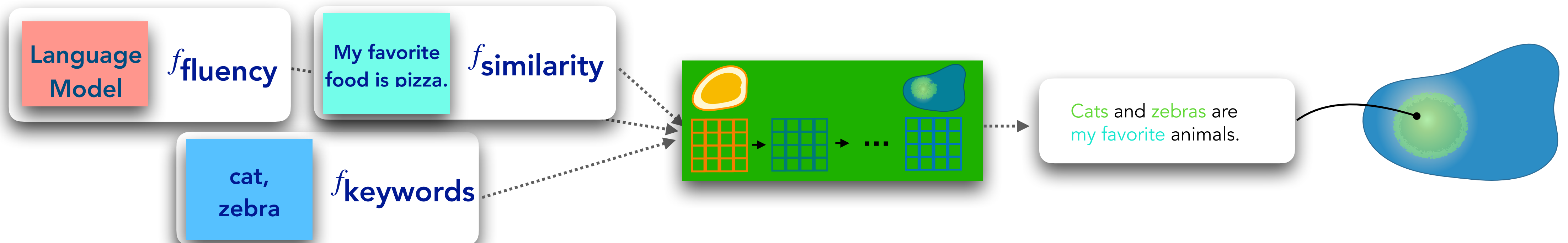
Models	Grammar	Left-coher. (x-y)	Right-coher. (y-z)	Overall-coher. (x-y-z)
COLD (Full)	4.17	3.96	2.88	2.83
COLD $- f_{\text{sim}}$	4.54	3.82	2.73	2.69
COLD $- f_{\text{LM}}^{\leftarrow}$	4.35	3.97	2.84	2.80
COLD $- f_{\text{pred}}$	4.61	4.07	2.75	2.77

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](https://arxiv.org/abs/2202.11705)

github.com/qkaren/COLD_decoding

Constrained generation

Looking ahead

Constrained generation

Looking ahead

- **Grounded generation**

Theorem

Let $M = (A, d)$ be a metric space.
Then M is a perfectly normal space.

Gold Proof

By definition, a topological space is perfectly normal space if and only if it is;
a perfectly T_4 space
a T_1 (Fréchet) space.

We have that:

- a Metric Space is Perfectly T_4
- a Metric Space is T_2 (Hausdorff)
- a T_2 (Hausdorff) Space is a T_1 (Fréchet) Space.

Computer-Generated Proof

From:

- Metric Space is Hausdorff
- T_2 (Hausdorff) Space is T_1 Space
- Metric Space is Perfectly T_4
- it follows that M is a topological space which is perfectly normal.

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

Constrained generation

Looking ahead

- **Grounded generation**

Theorem

Let $M = (A, d)$ be a metric space.
Then M is a perfectly normal space.

Gold Proof

By definition, a topological space is perfectly normal if and only if it is:
a perfectly T_4 space
a T_1 (Fréchet) space.

We have that:

- a Metric Space is Perfectly T_4
- a Metric Space is T_2 (Hausdorff)
- a T_2 (Hausdorff) Space is a T_1 (Fréchet) Space.

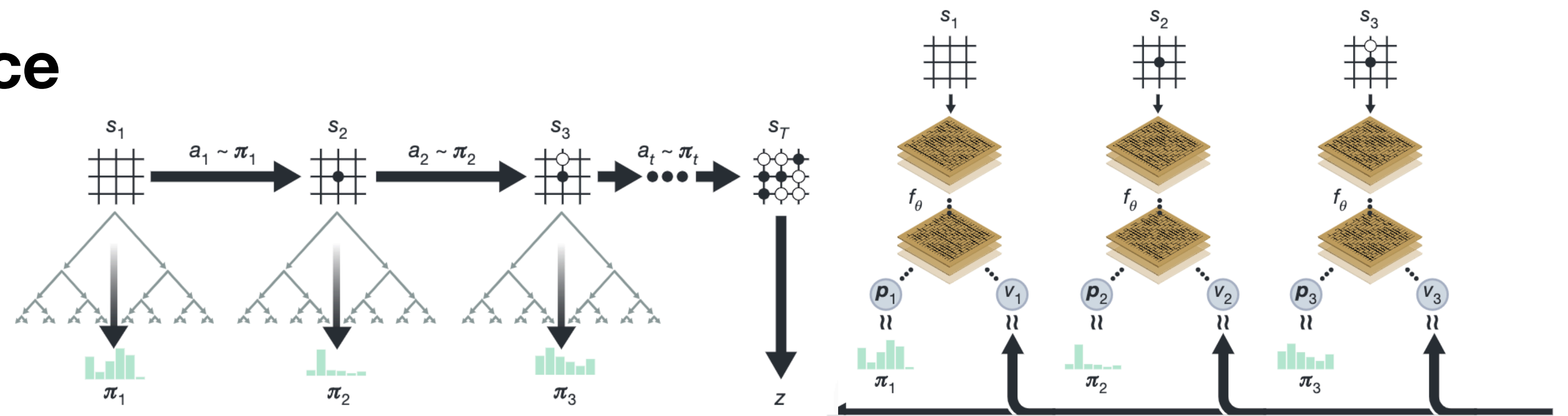
Computer-Generated Proof

From:

- Metric Space is Hausdorff
- T_2 (Hausdorff) Space is T_1 Space
- Metric Space is Perfectly T_4
- it follows that M is a topological space which is perfectly normal.

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

- **Joint learning & inference**



Thanks for your attention!