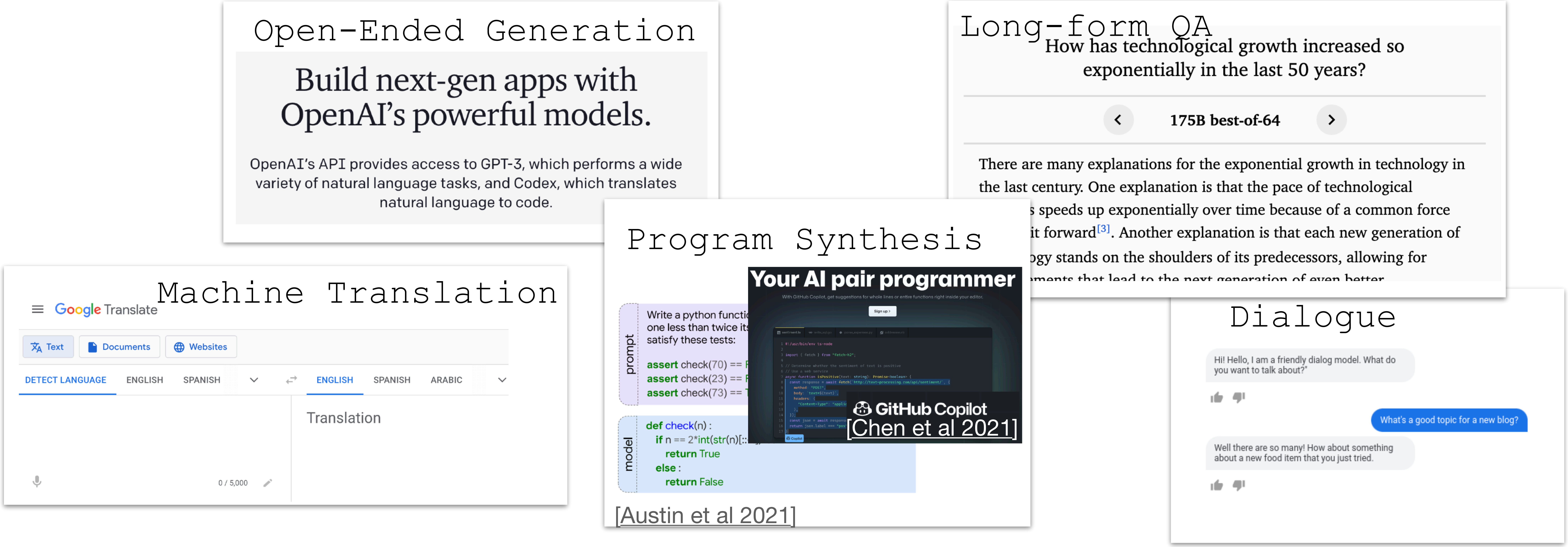
Constrained text generation through discrete & continuous inference

Sean Welleck | 05.04.2022

UNIVERSITY of WASHINGTON

Neural text generation

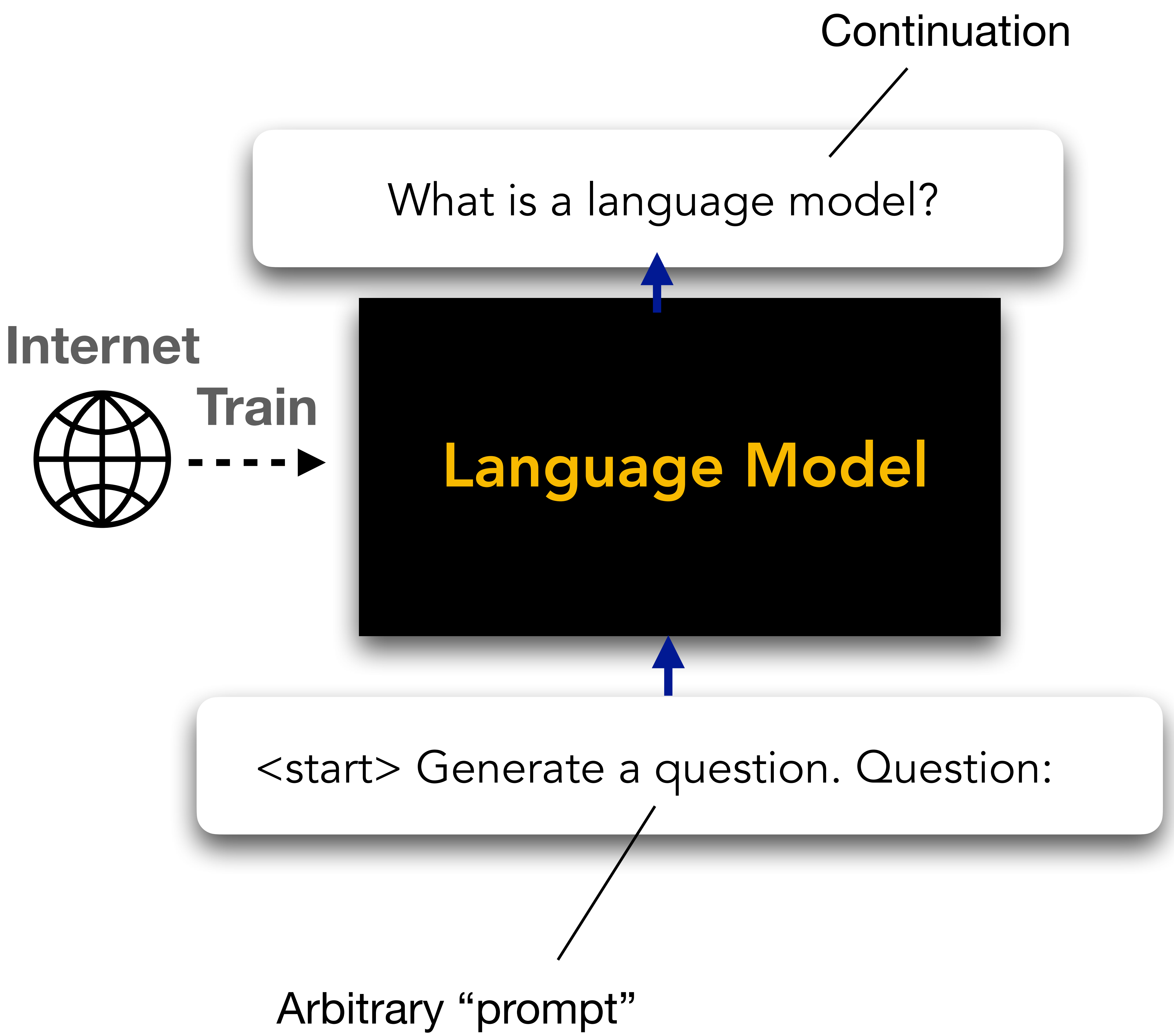
- **Large-scale language models** drive state-of-the-art performance in text generation tasks:



[Thoppilan et al 2022]

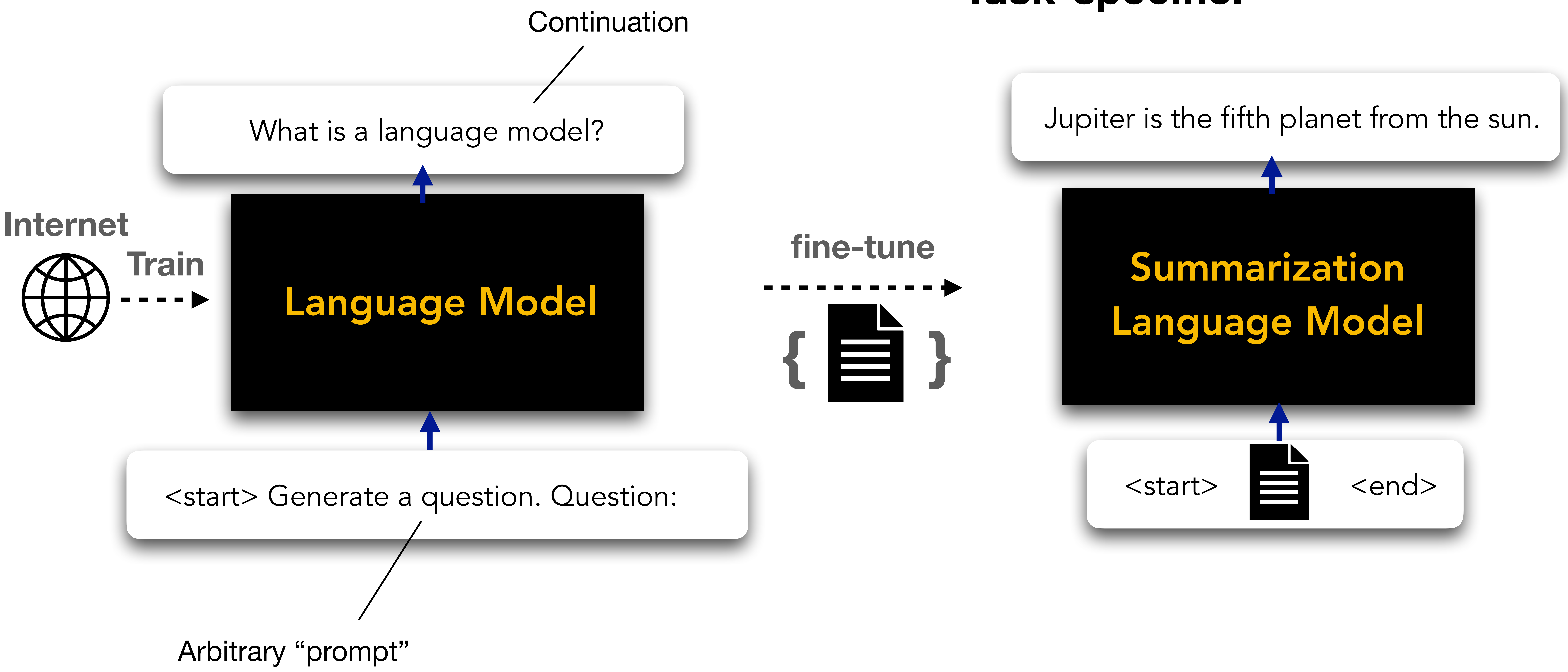
Neural text generation

- **General purpose:**

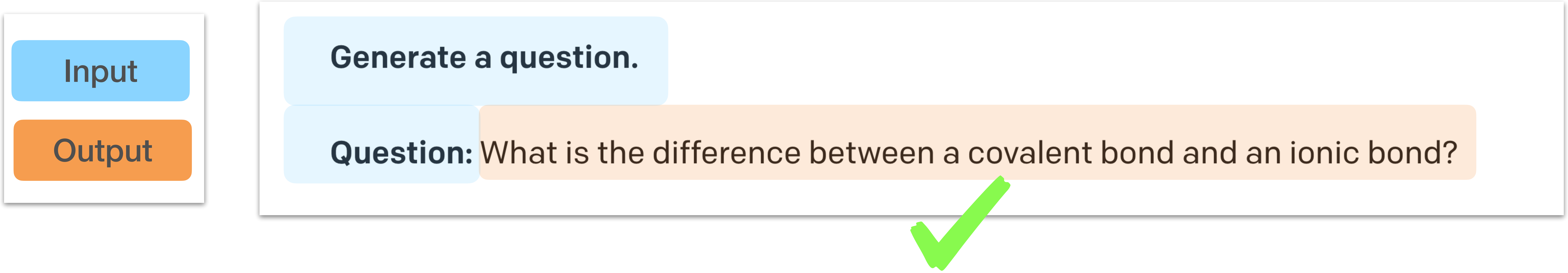


Neural text generation

- **General purpose:**
- **Task-specific:**



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



Example from: https://beta.openai.com/playground

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

Input

Summarize this for a second-grade student:

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 size.

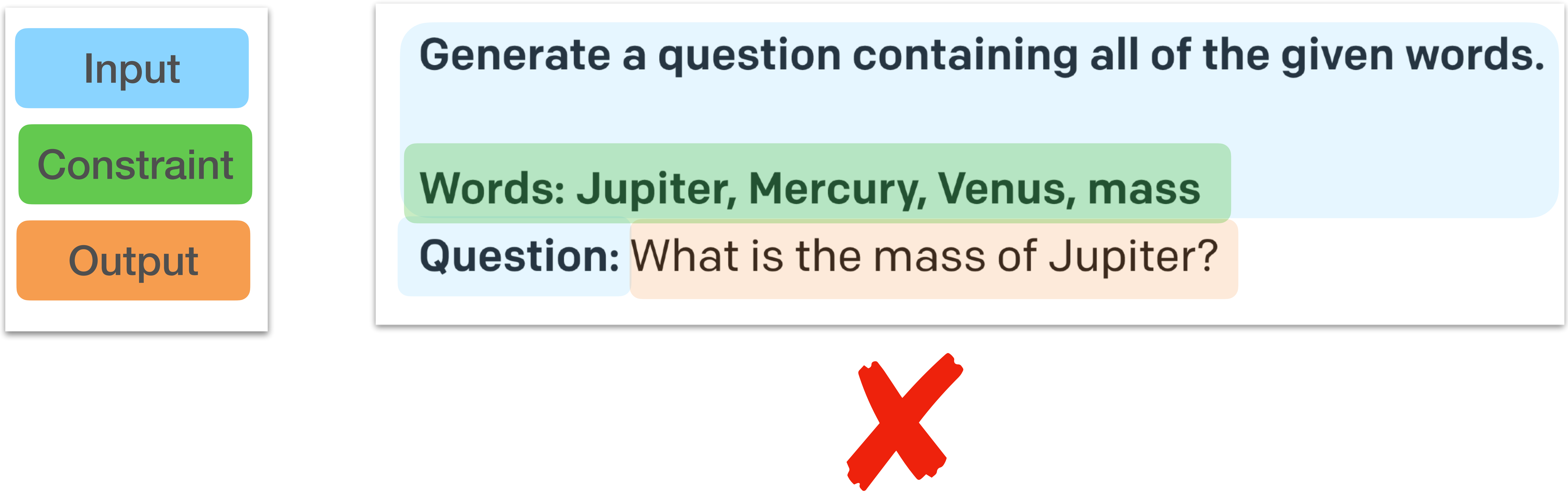
Output

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.

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
  - Lexical content



Example from: https://beta.openai.com/playground

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 size.

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.

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

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?

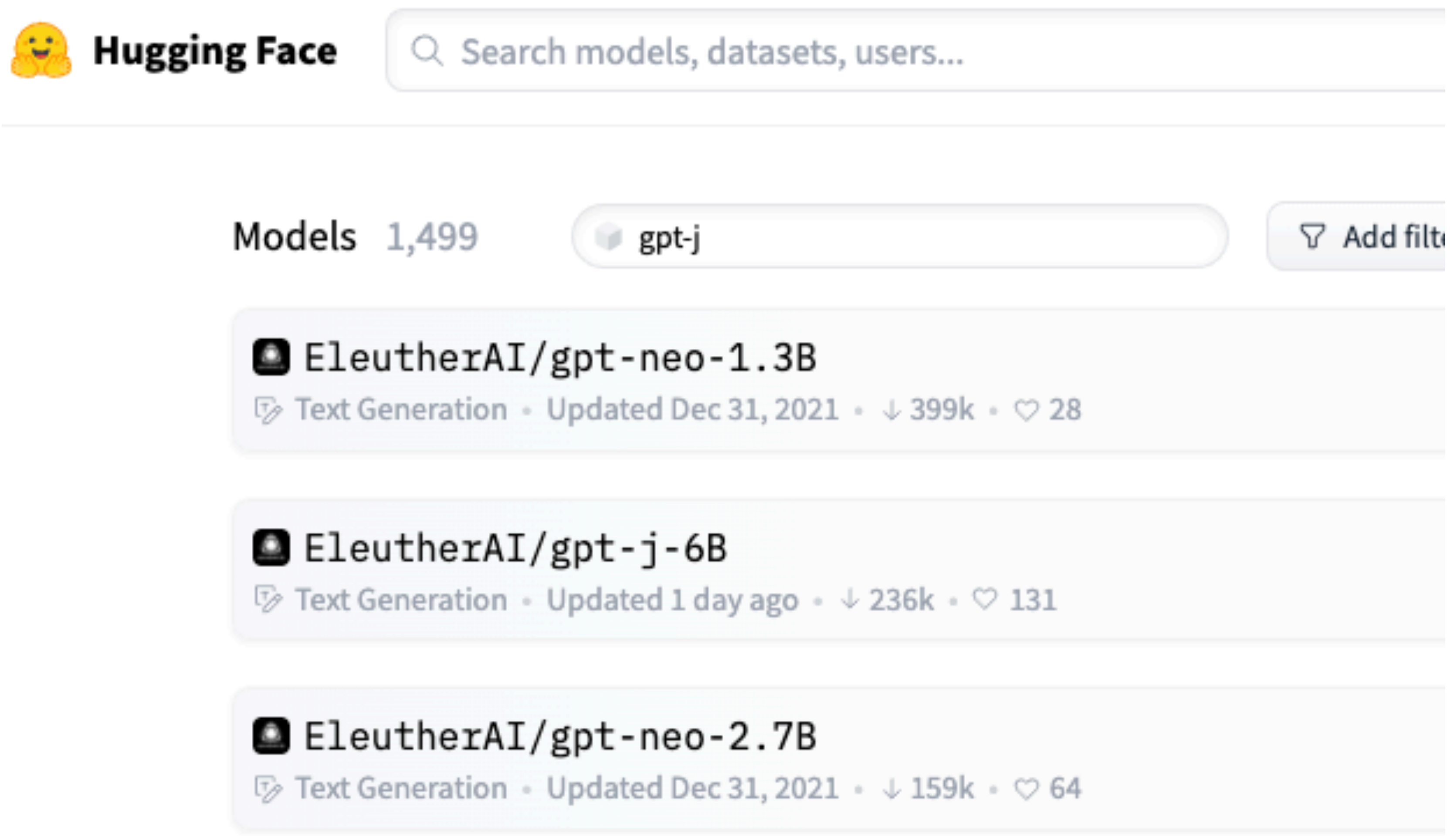
Output

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.

Example based on: https://beta.openai.com/playground/p/default-summarize

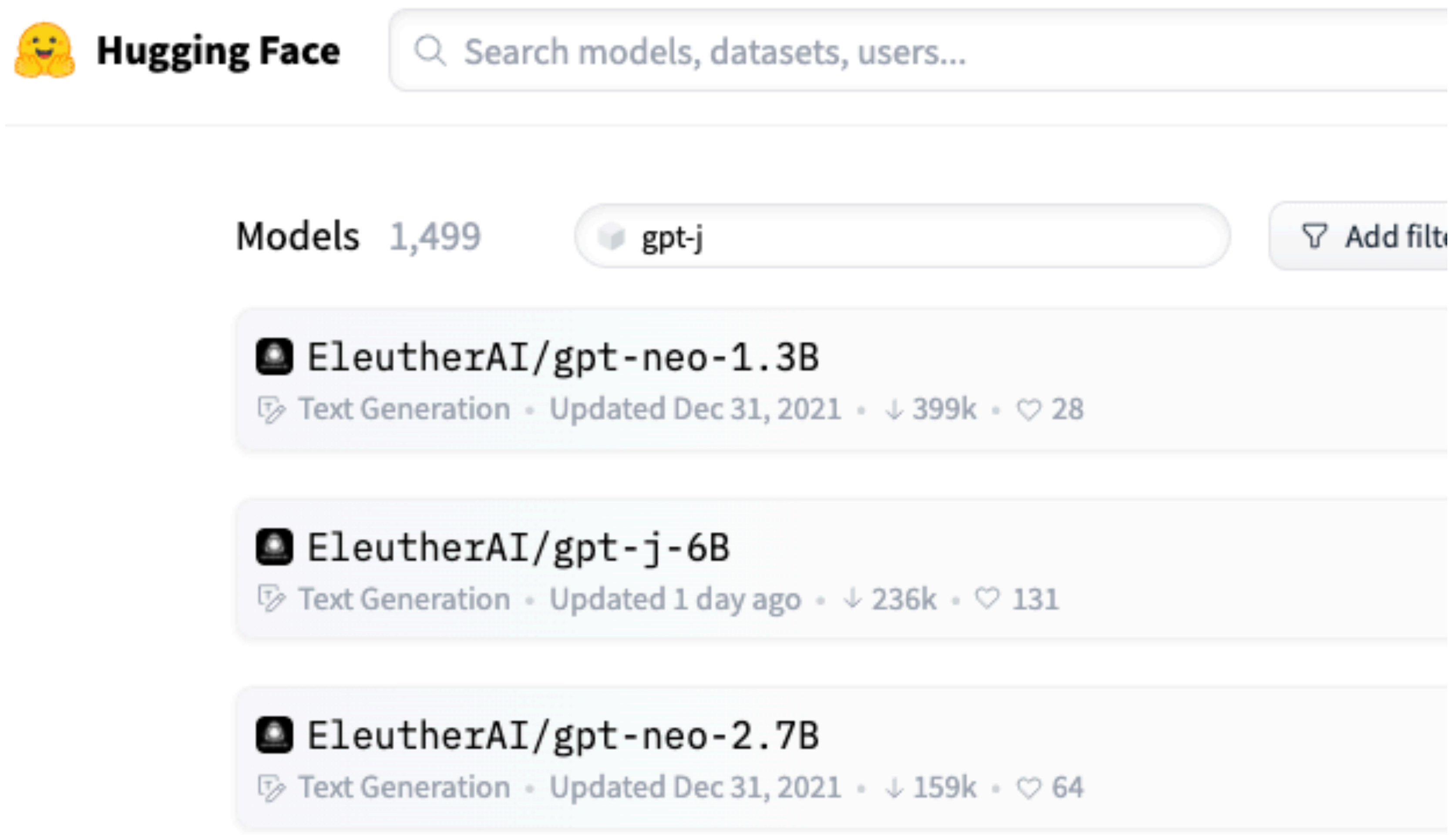
Controlling neural text generation

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



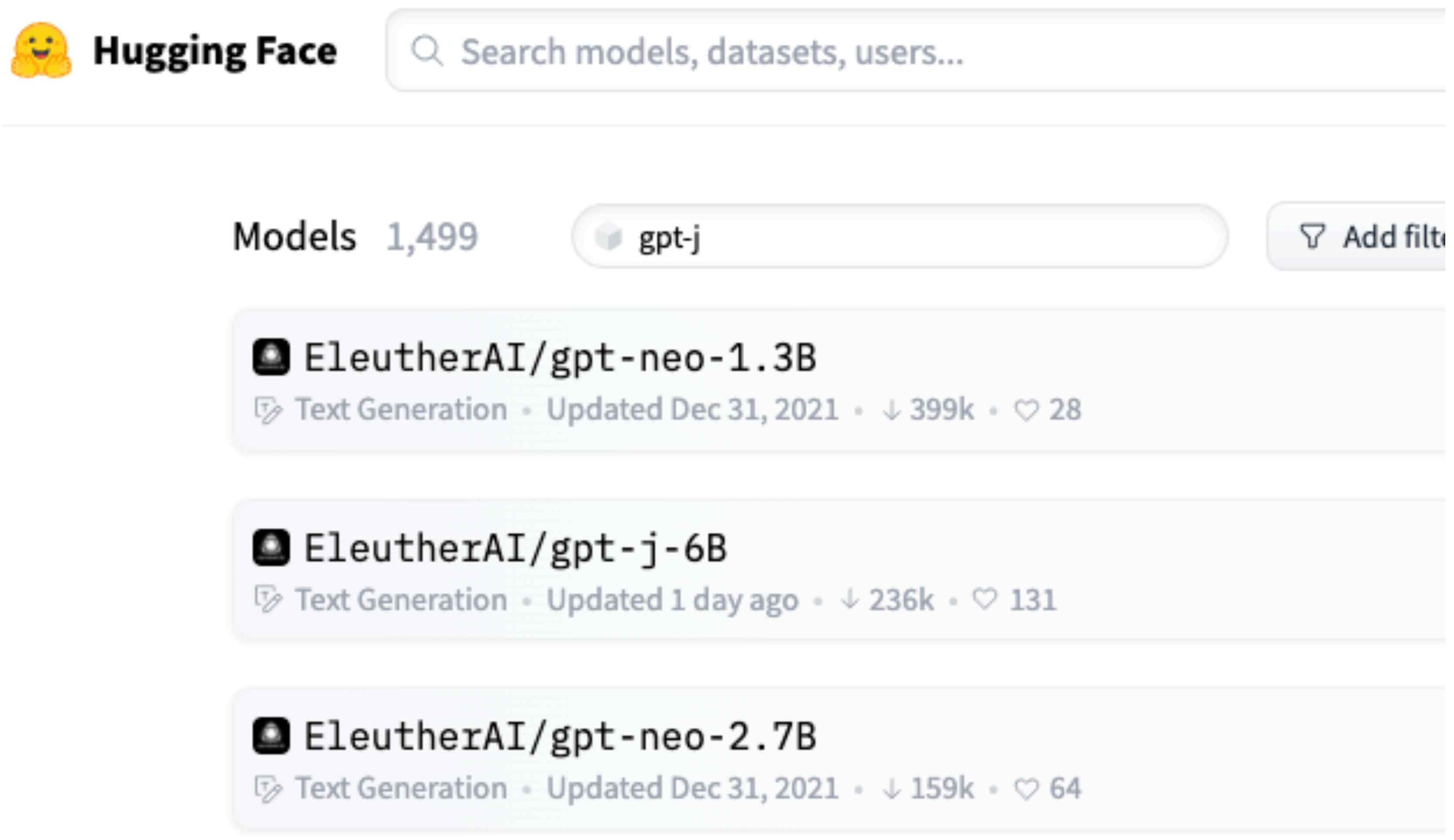
Controlling neural text generation

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



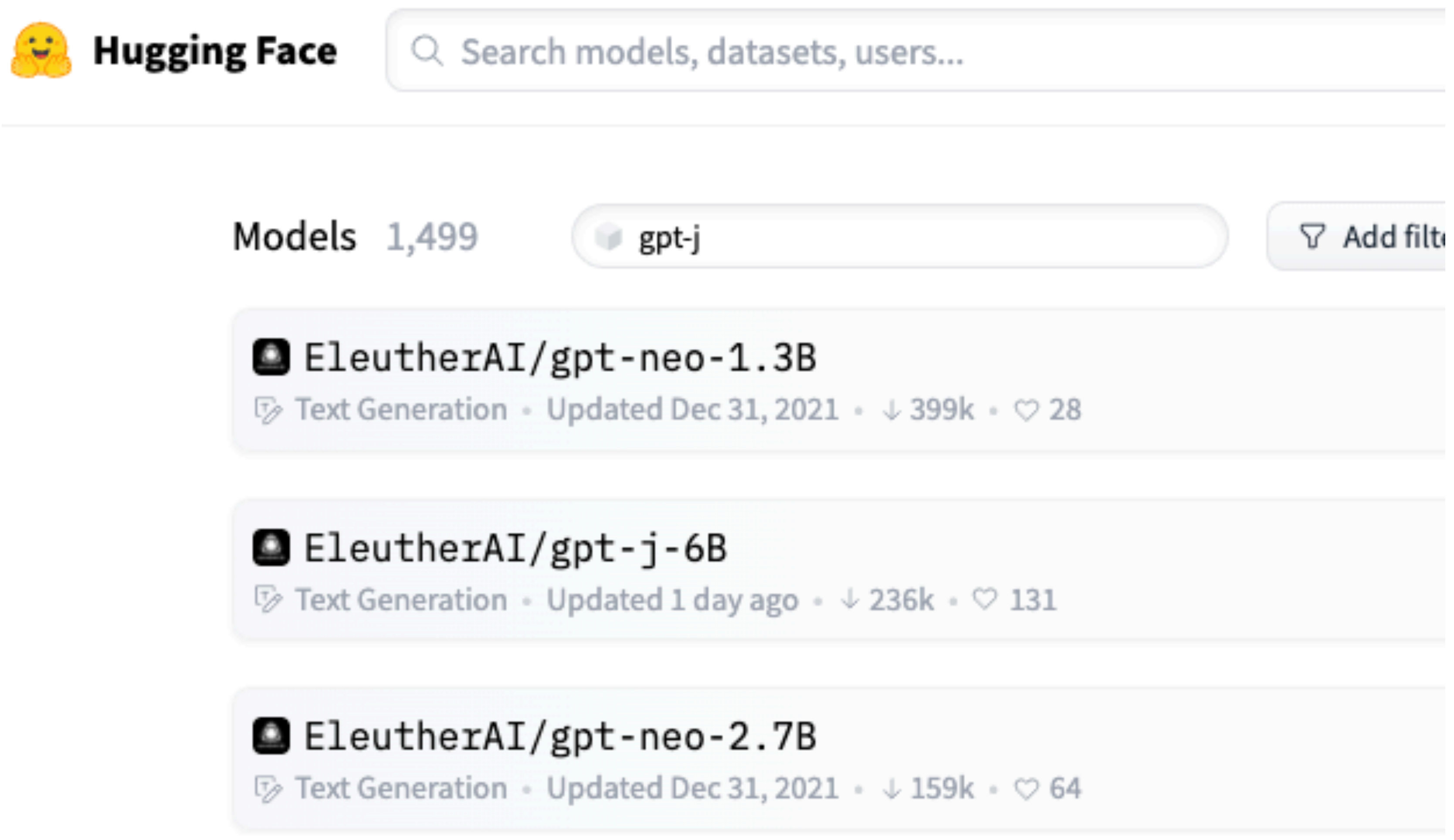
Controlling neural text generation

- 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



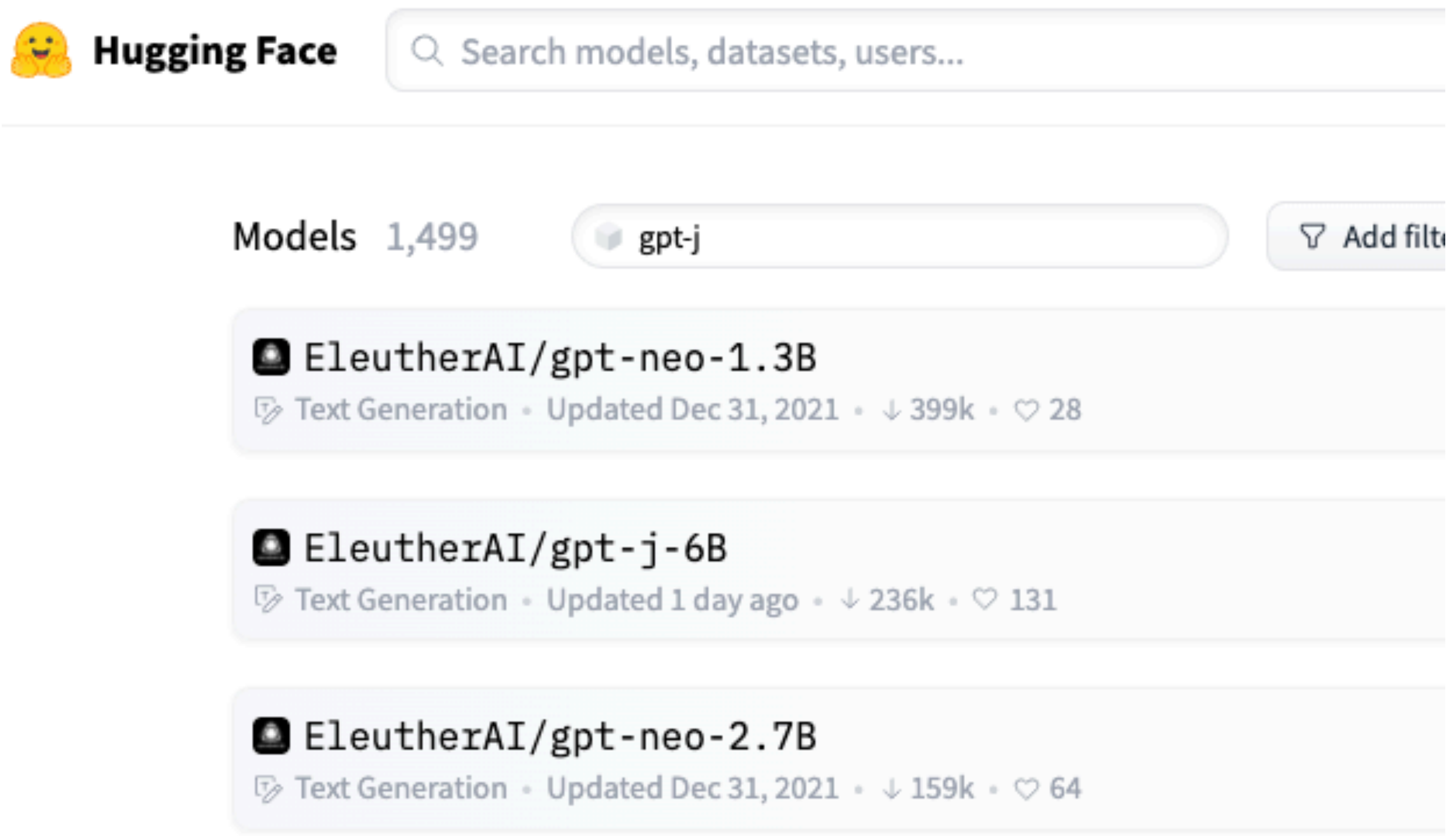
Controlling neural text generation

- 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

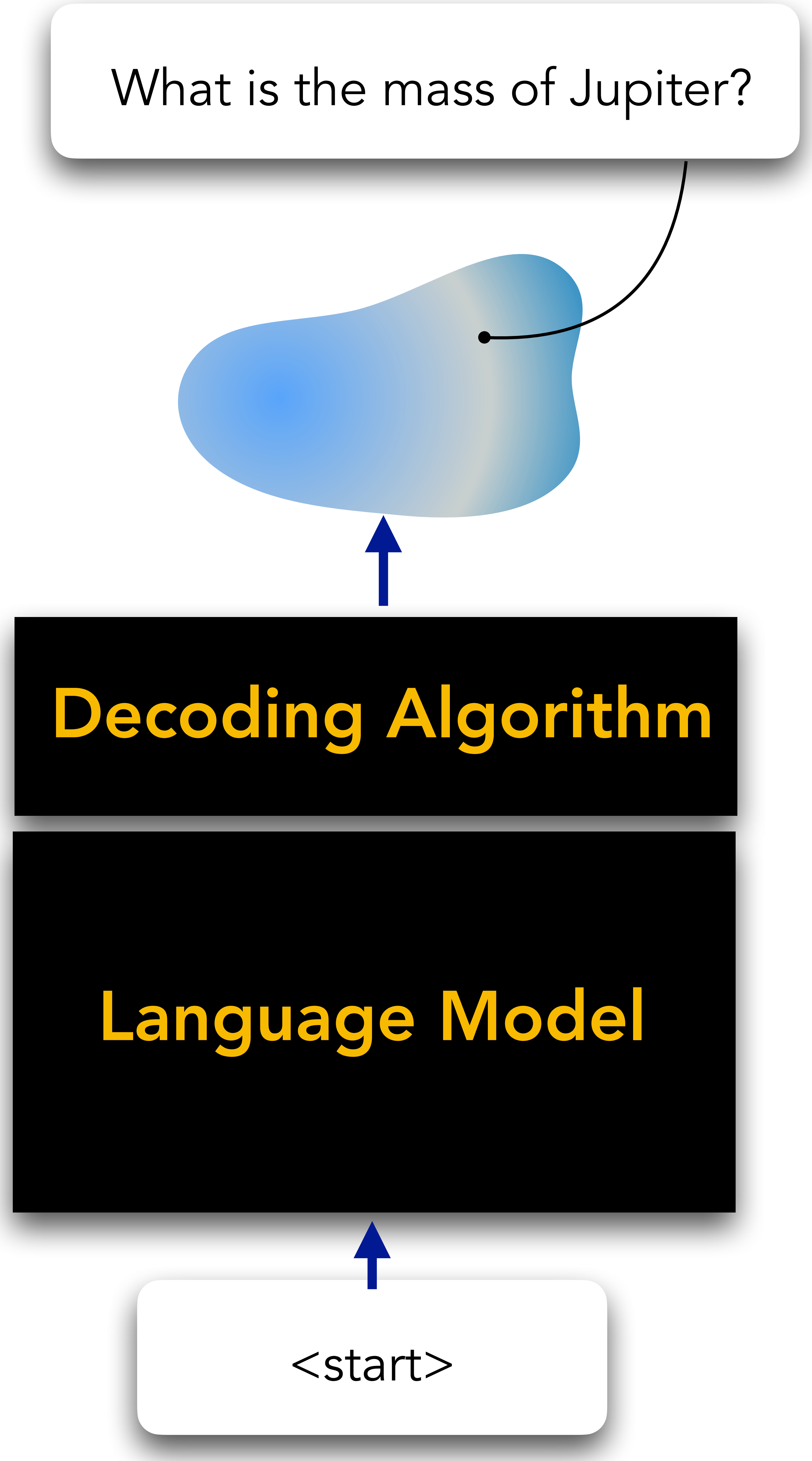
- Text generation involves two steps:
- Learn a **model** from data (or download one…)
- $p_\theta(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, \mathbf{x})$

Language Model

Control through inference

Model + decoding

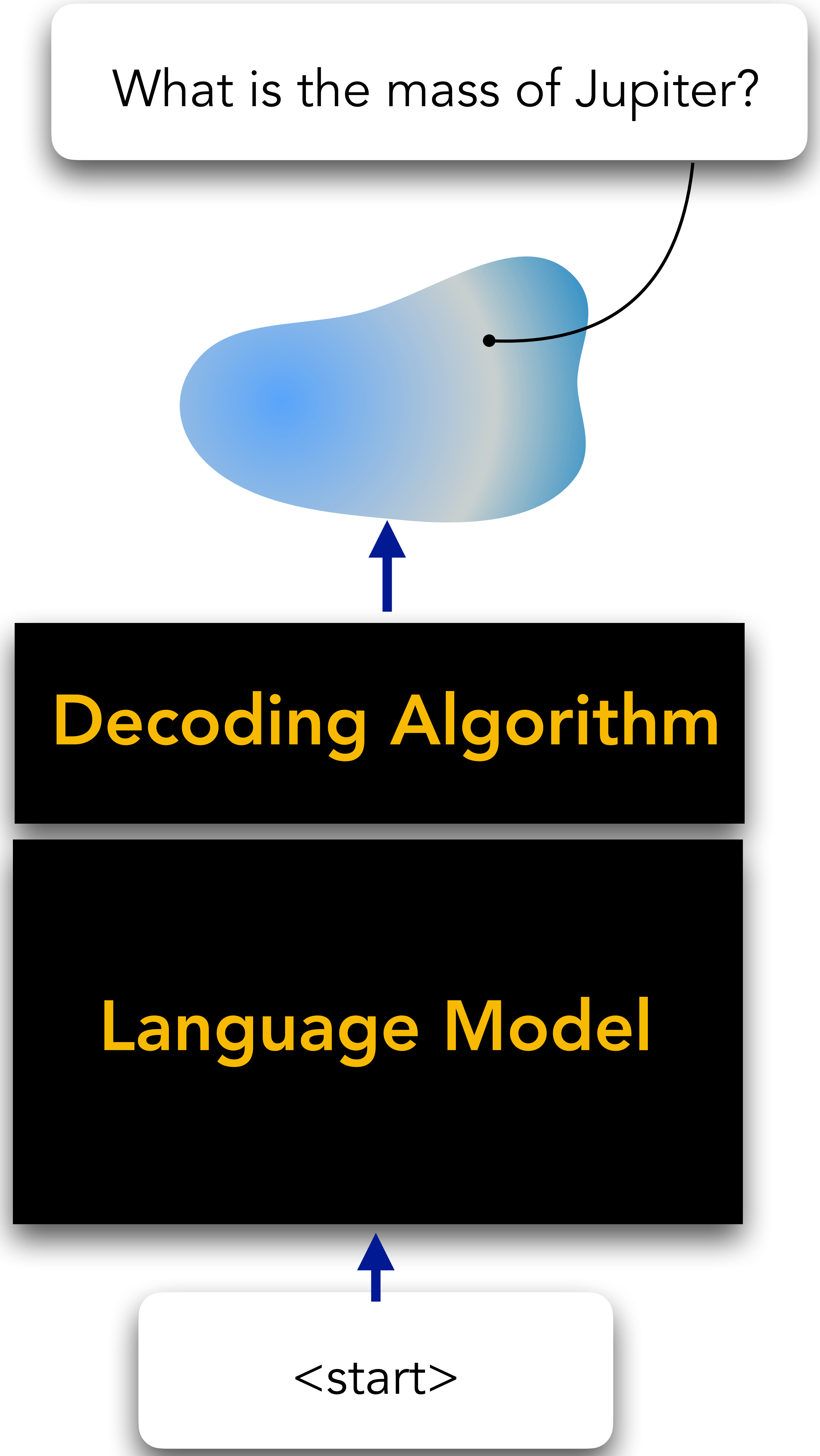
- Text generation involves two steps:
- Learn a **model** from data (or download one…)
    - $p_\theta(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, \mathbf{x})$
- Use an **inference/decoding** algorithm to generate text
    - $\hat{\mathbf{y}} = \text{decode}(p_\theta(\cdot \mid \mathbf{x}))$



Control through inference

Model + decoding

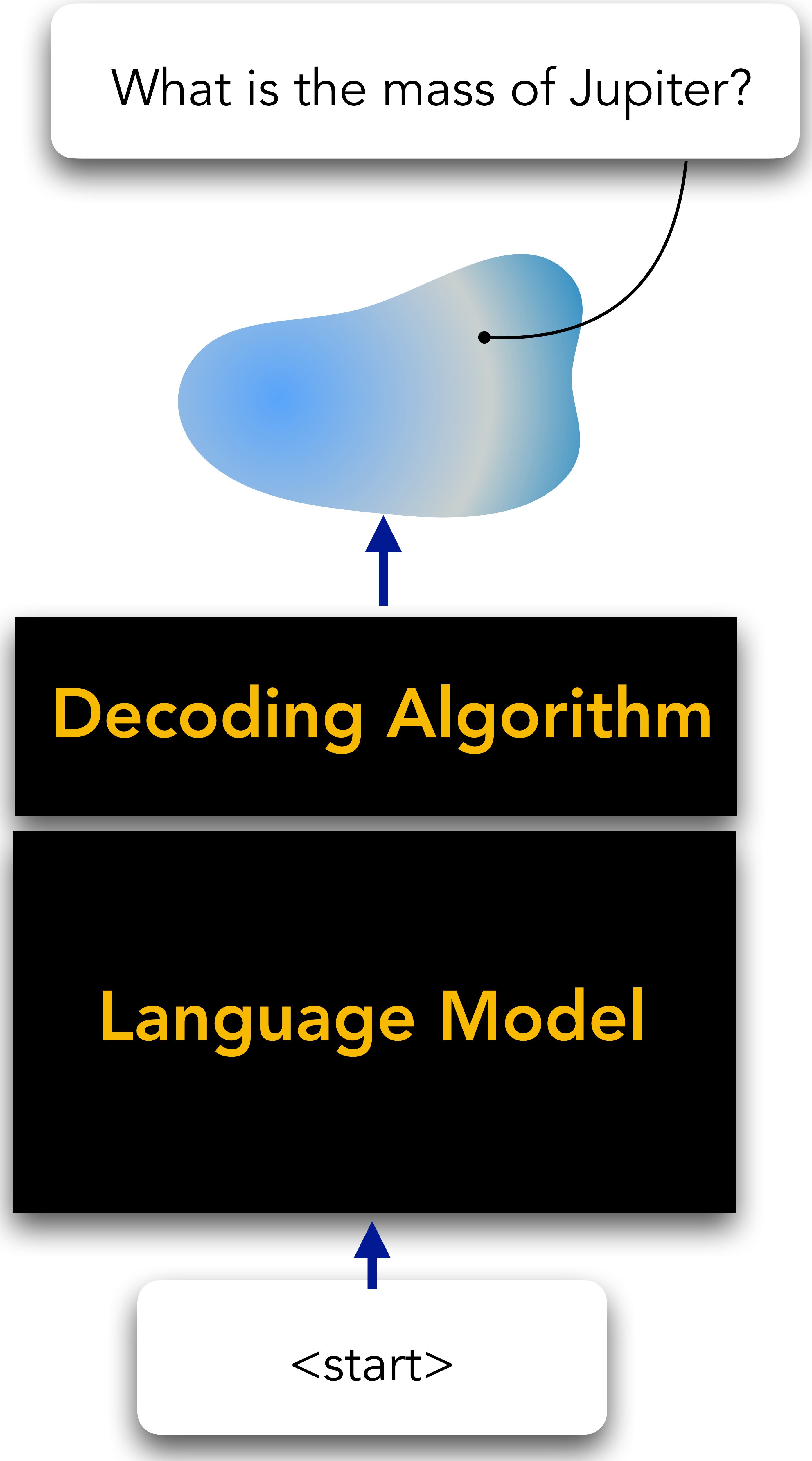
- Text generation involves two steps:
- Learn a **model** from data (or download one…)
  - $p_\theta(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, \mathbf{x})$
- Use an **inference/decoding** algorithm to generate text
  - $\hat{\mathbf{y}} = \text{decode}(p_\theta(\,\cdot\mid \mathbf{x}))$
    - e.g. sampling, $\mathbf{y_t} \sim p_\theta(y_t \mid y_{<t}, \mathbf{x})$



Control through inference

Model + decoding

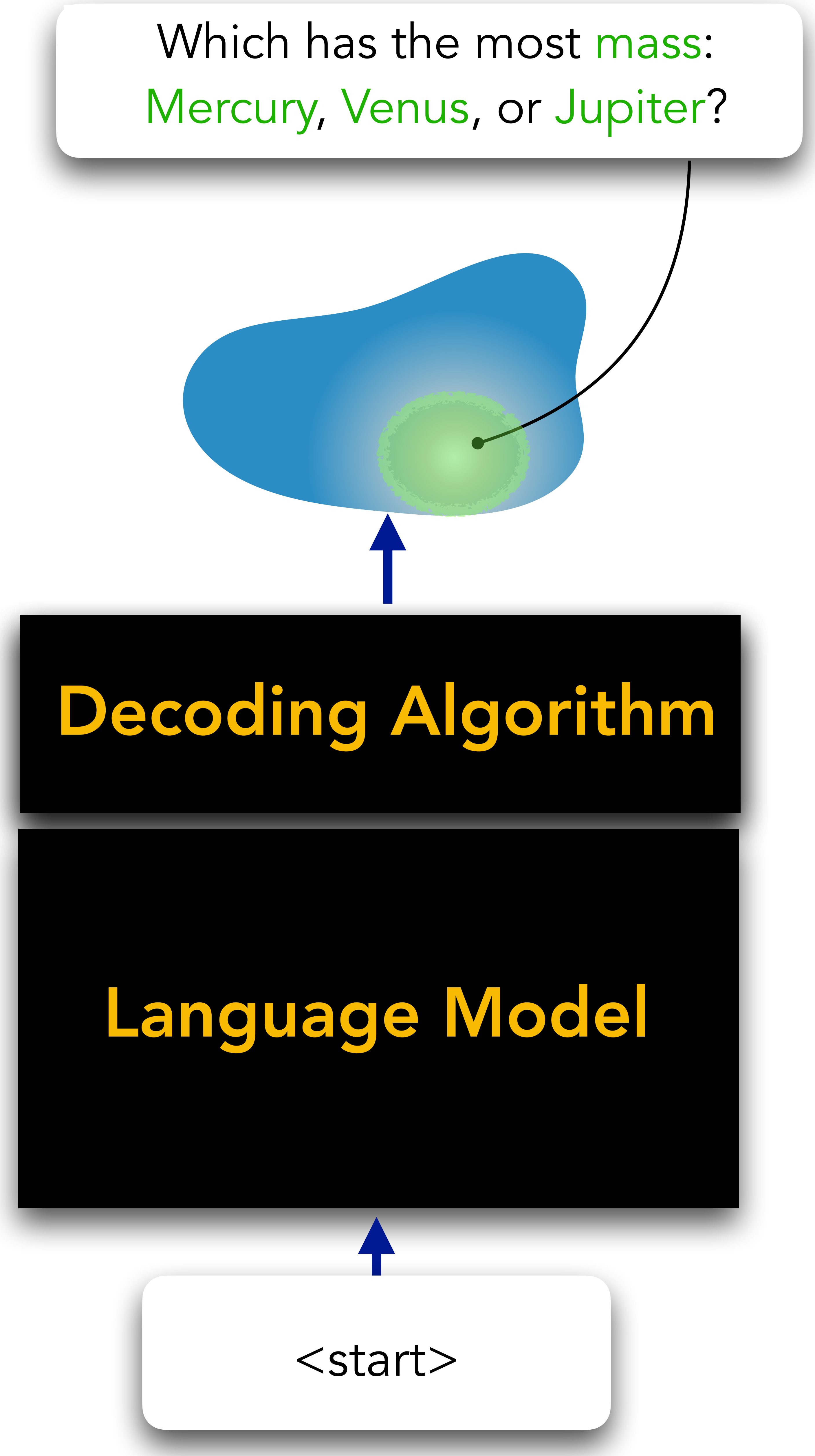
- Text generation involves two steps:
- Learn a **model** from data (or download one…)
- $p_\theta(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, \mathbf{x})$
- Use an **inference/decoding** algorithm to generate text
  - $\hat{\mathbf{y}} = \text{decode}(p_\theta(\cdot \mid \mathbf{x}))$
    - e.g. sampling, $\mathbf{y_t} \sim p_\theta(y_t \mid y_{<t}, \mathbf{x})$
    - e.g. maximization $y_t = \arg\max_{y_t} p_\theta(y_t \mid y_{<t}, \mathbf{x})$



Constraints through inference

Model + decoding

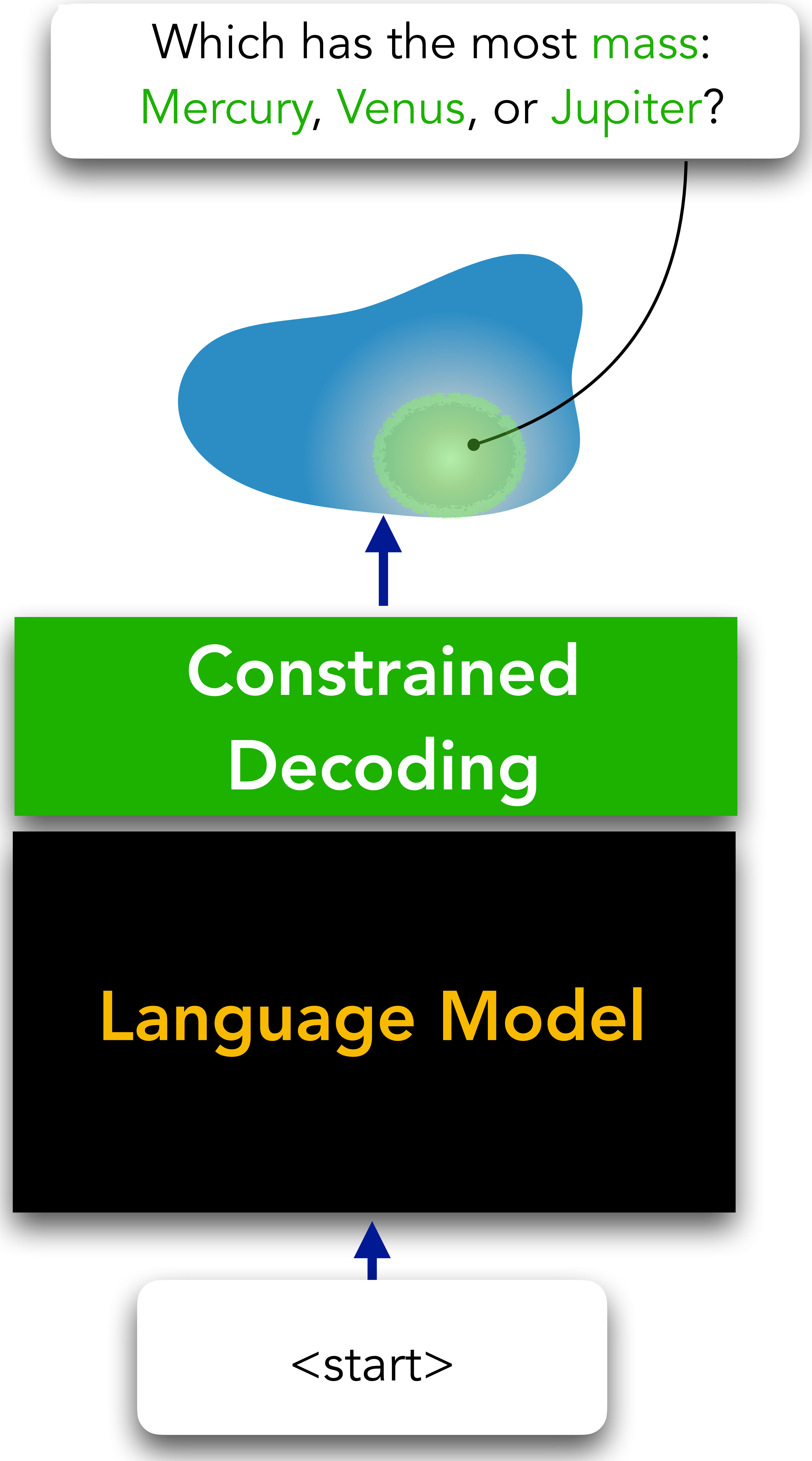
- Control: **constraints** on the generation distribution



Constraints through inference

Model + decoding

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

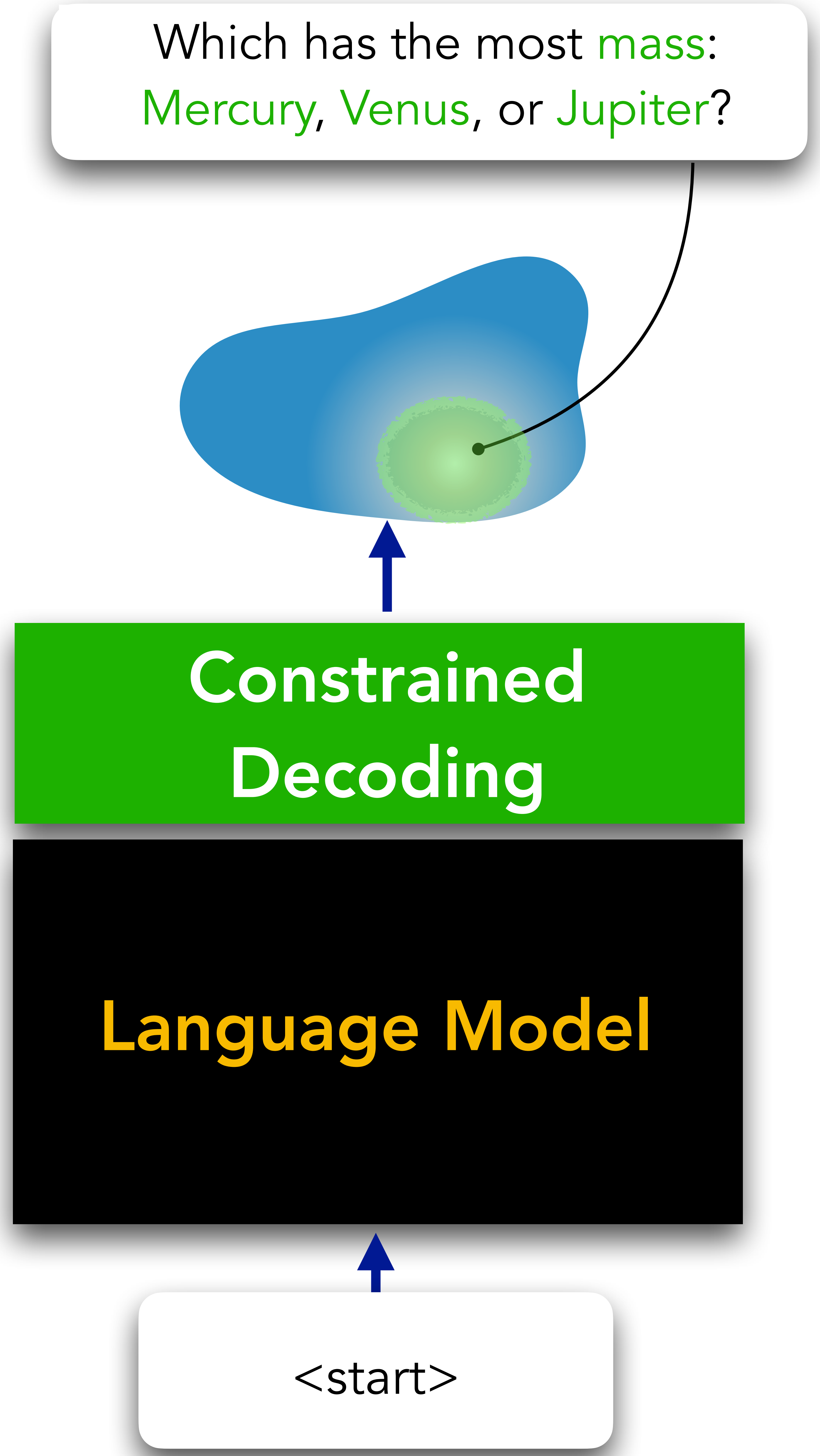


Constraints through inference

Model + decoding

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

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

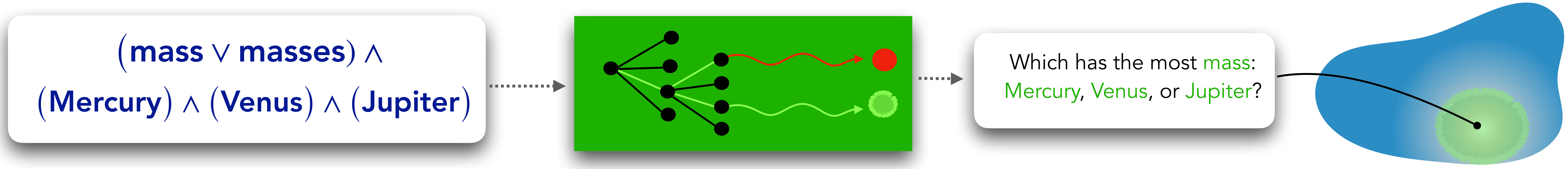


Constrained generation through inference

- Today: decoding algorithms for constrained generation from two perspectives

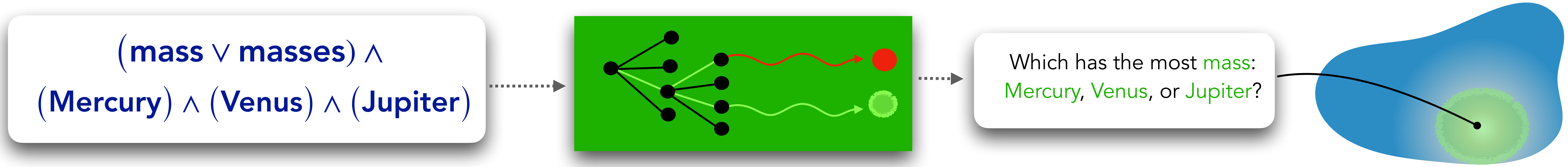
Constrained generation through inference

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

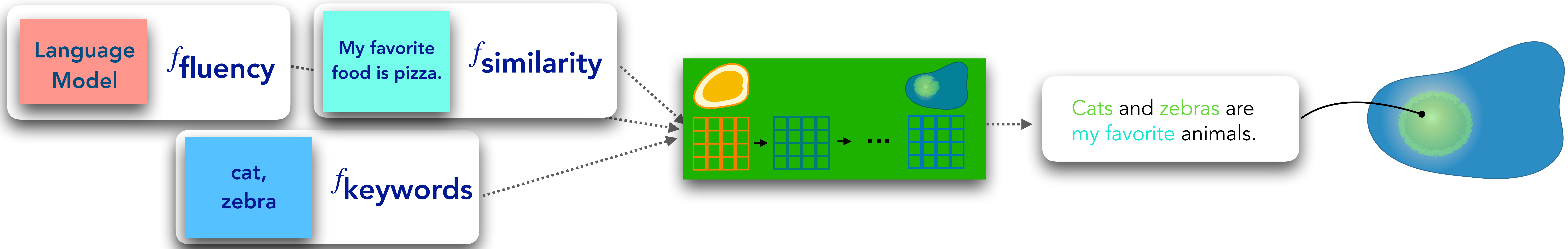


Constrained generation through inference

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



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



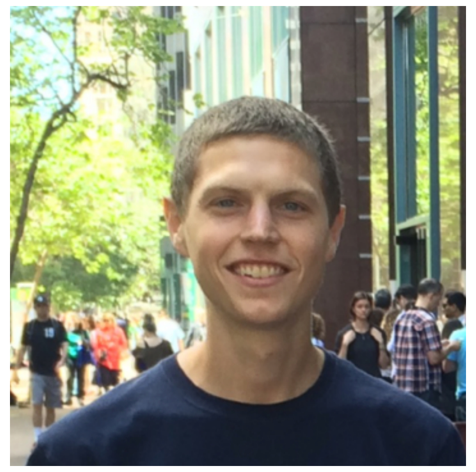
Constrained generation through *discrete* inference

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

NAACL 2022

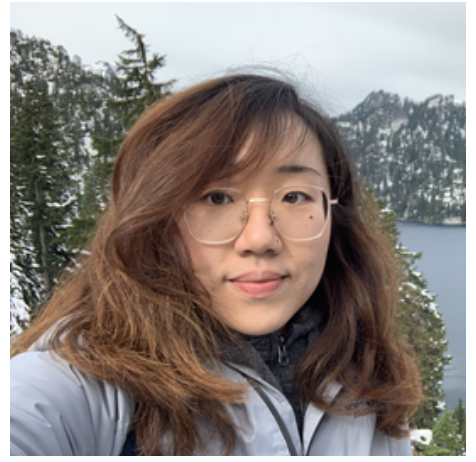
Ximing Lu

Sean Welleck



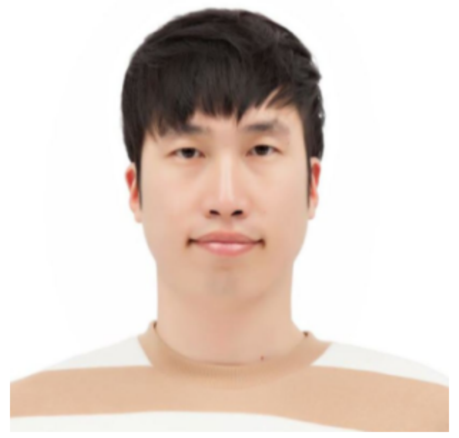
Peter West

Liwei Jiang



Lianhui Qin

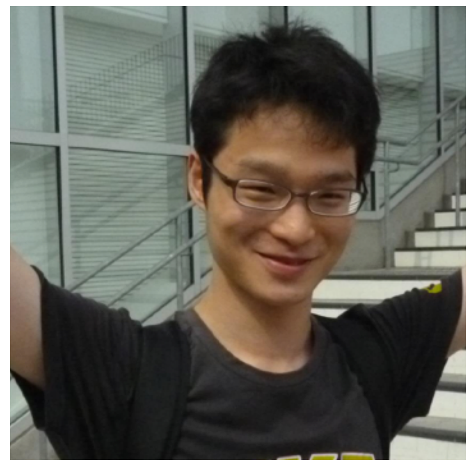
Youngjae Yu



Yejin Choi

Daniel Khashabi

Jungo Kasai



Ronan Le Bras

Rowan Zellers

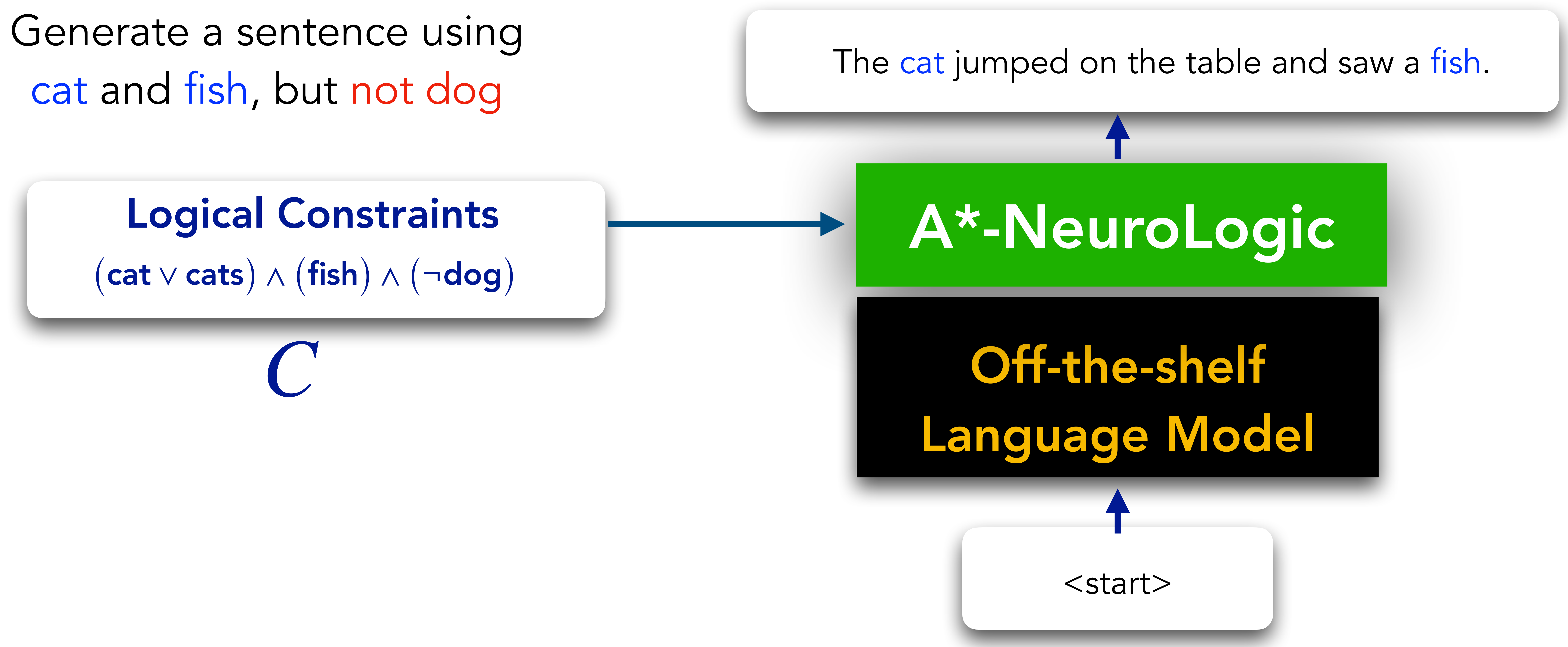
Noah Smith

W UNIVERSITY of WASHINGTON

Ai2

Logical lexical constraints

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



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

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

Standard decoding

Beam search

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

<s>

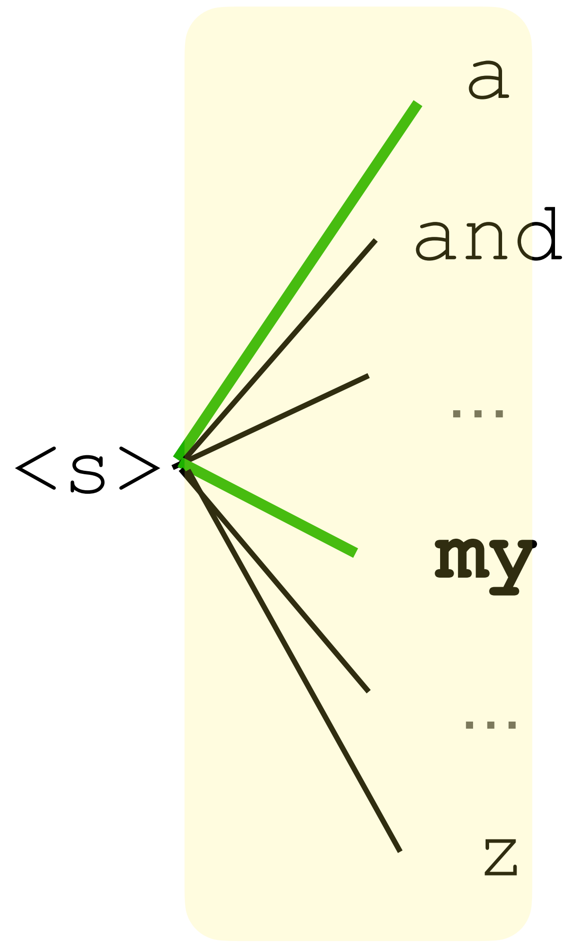
- Left-to-right search on the lattice of tokens:

Standard decoding

Beam search

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

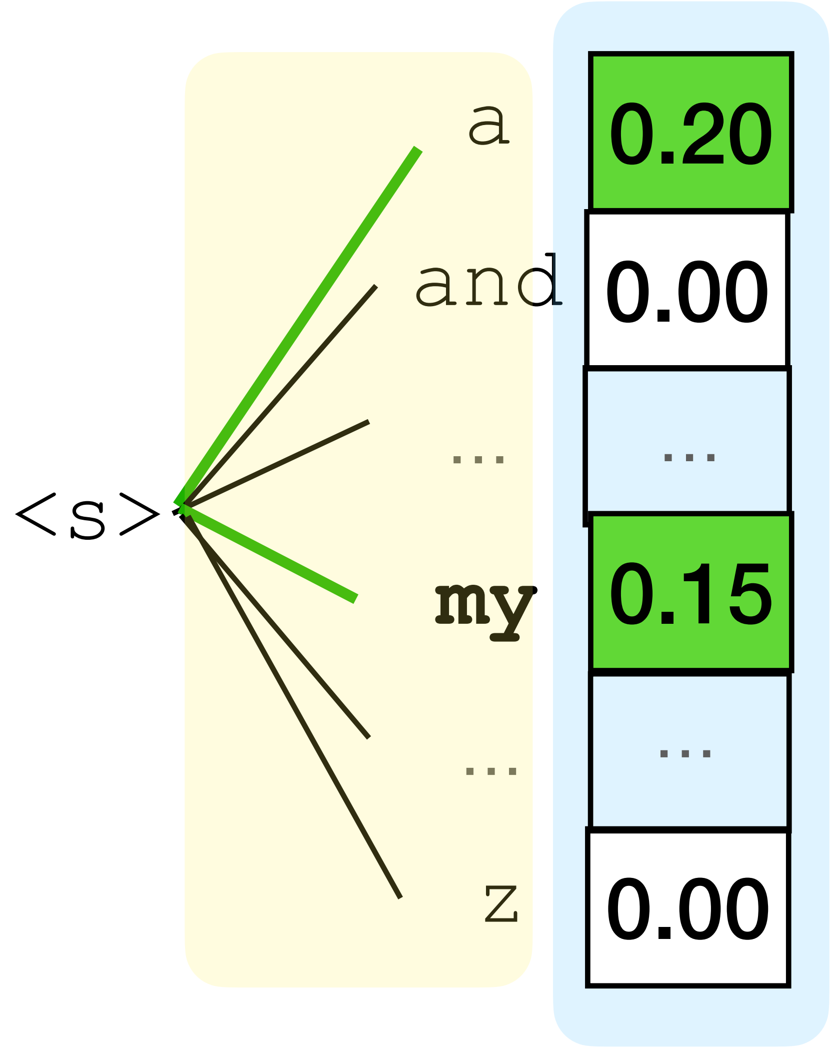
- Left-to-right search on the lattice of tokens:
  - Expand prefixes with next-tokens



Standard decoding

Beam search

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

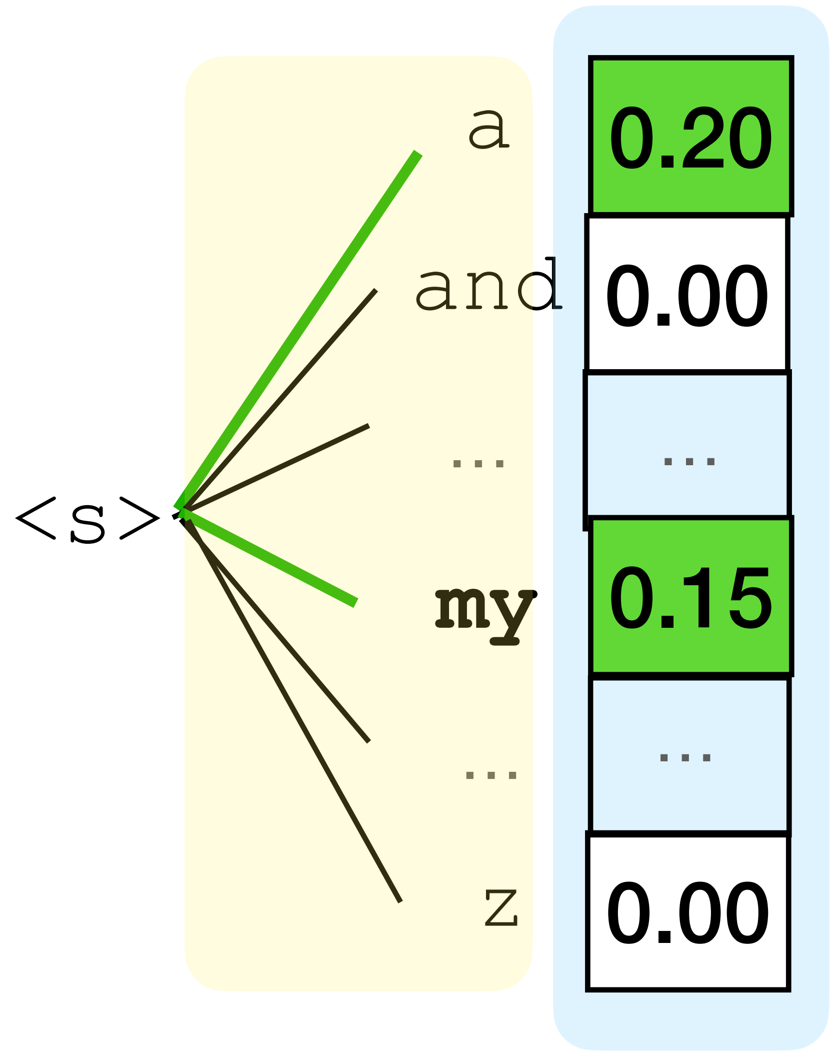


Standard decoding

Beam search

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

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

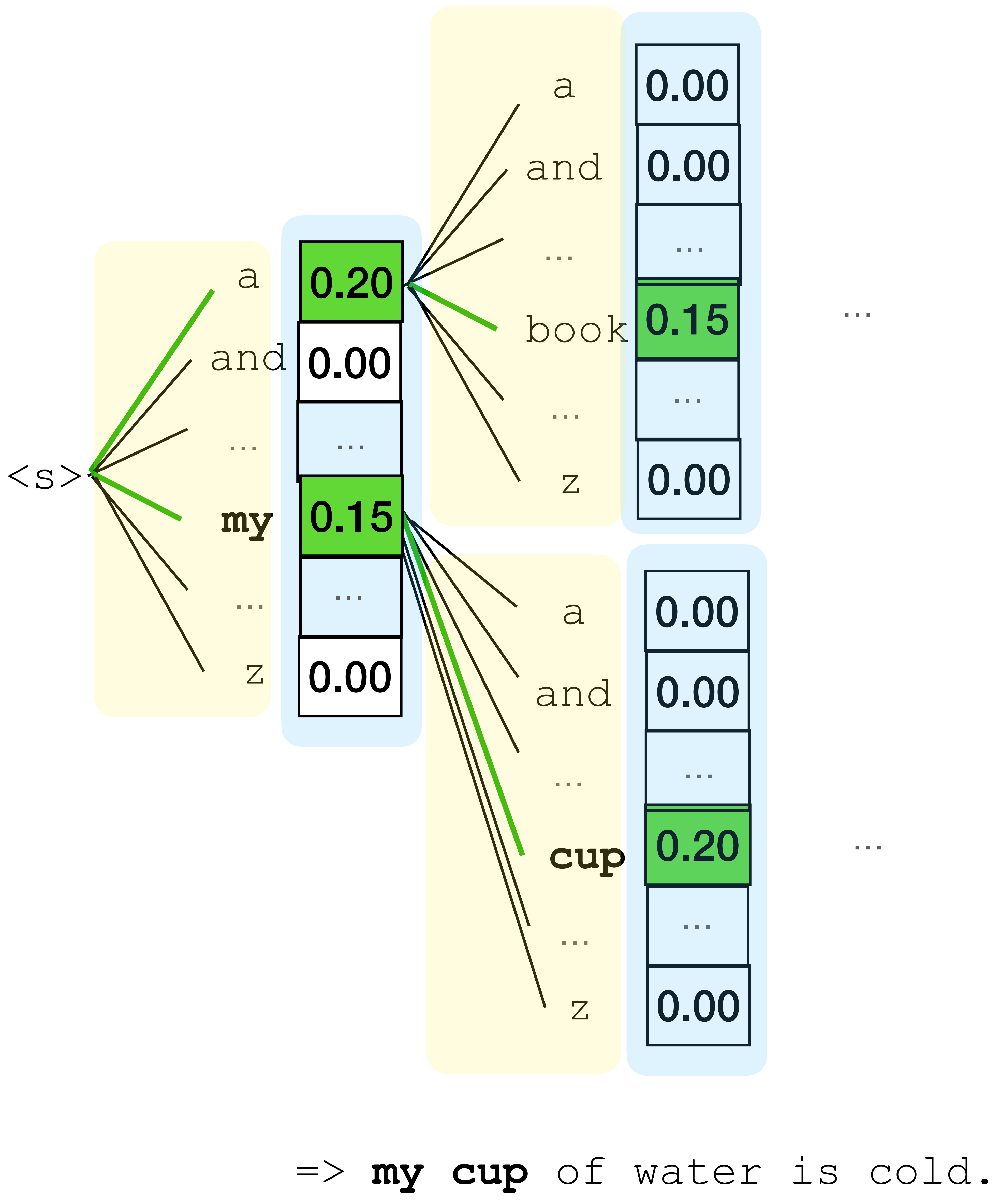


Standard decoding

Beam search

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

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

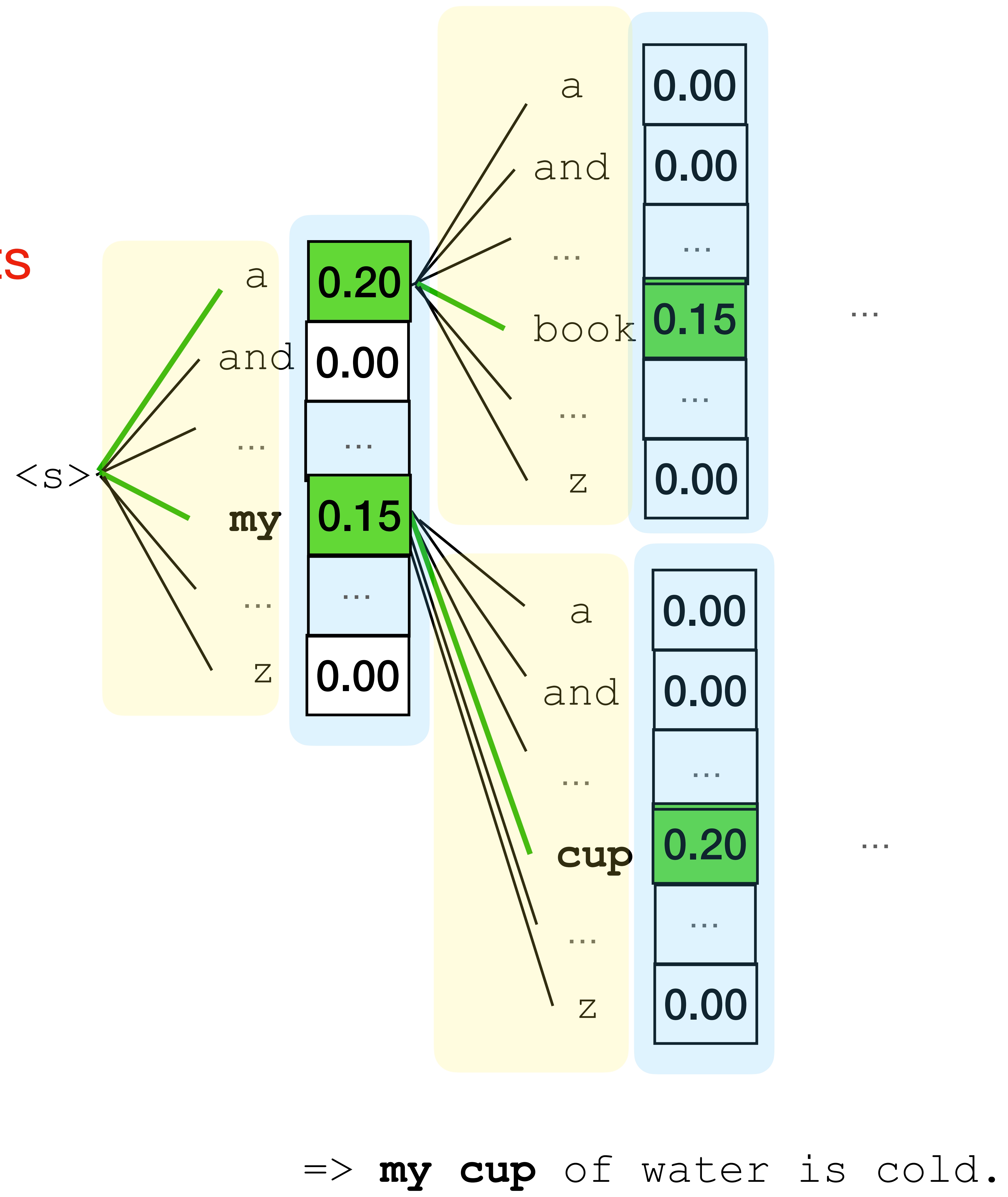


Standard decoding

Beam search

Ignores constraints

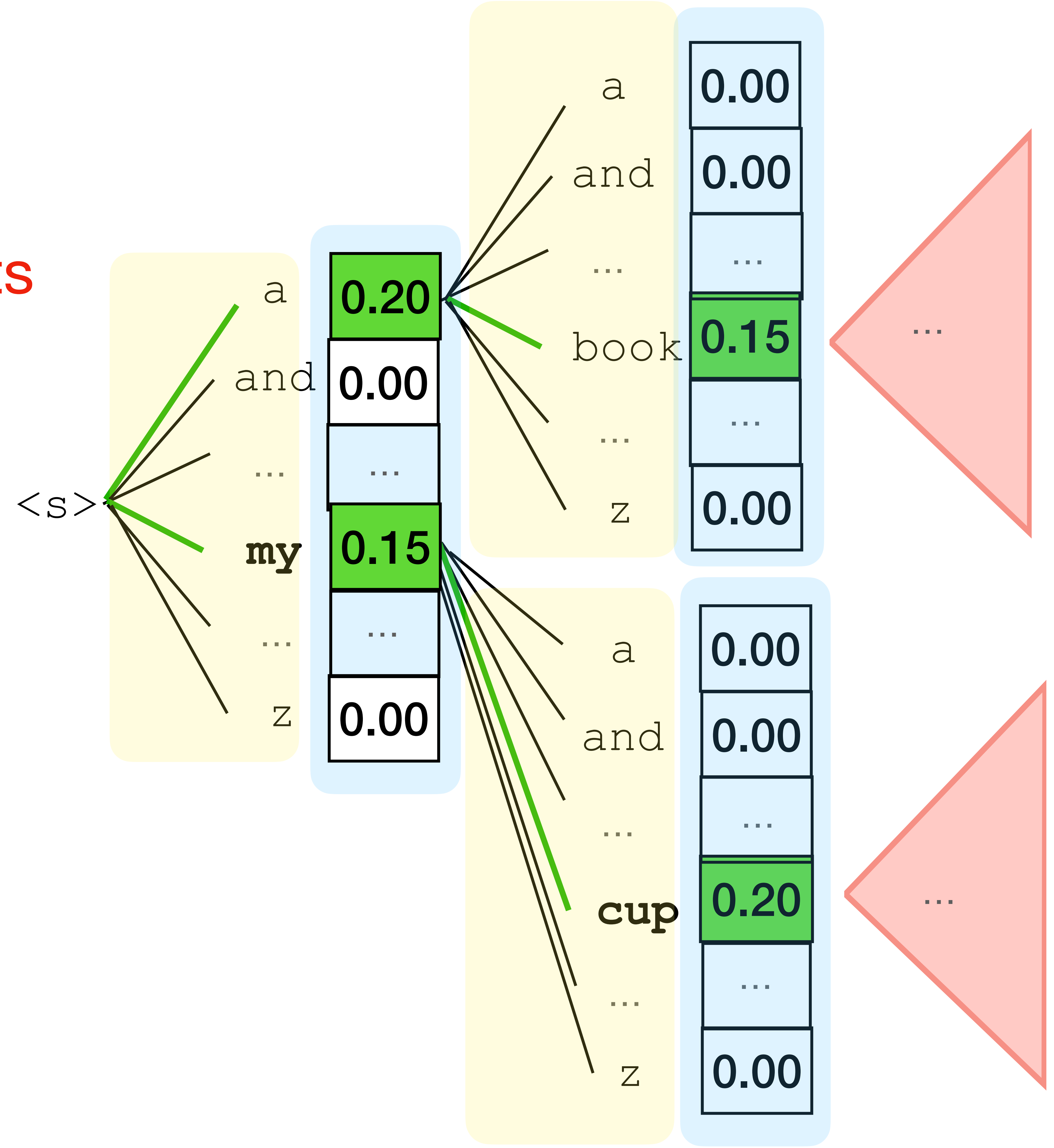
- $$\mathbf{y}_* \approx \arg\max_{\mathbf{y}\in\mathcal{Y}} \underbrace{\log p_\theta(\mathbf{y})}_{\text{fluency}} + \underbrace{0}_{\text{constraints}}$$
- Left-to-right search on the lattice of tokens:
  - Expand prefixes with next-tokens
  - Score each using $\underbrace{\log p_\theta(y_t \mid y_{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



Standard decoding

Beam search

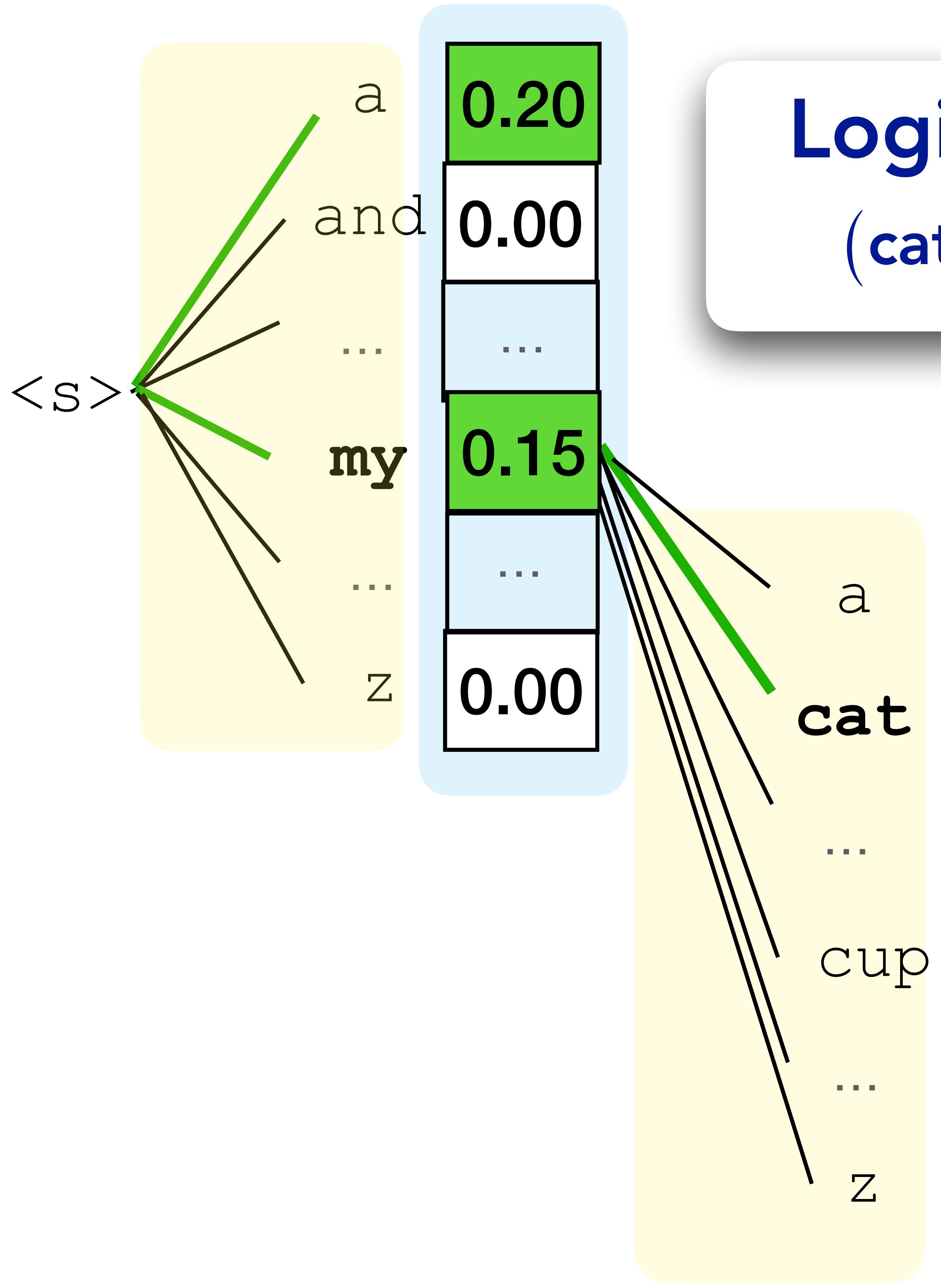
- $$\mathbf{y}_* \approx \arg\max_{\mathbf{y}\in\mathcal{Y}} \underbrace{\log p_\theta(\mathbf{y})}_{\text{fluency}} + \underbrace{0}_{\text{constraints}}$$ Ignores constraints
- Left-to-right search on the lattice of tokens:
  - Expand prefixes with next-tokens
  - Score each using $\underbrace{\log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



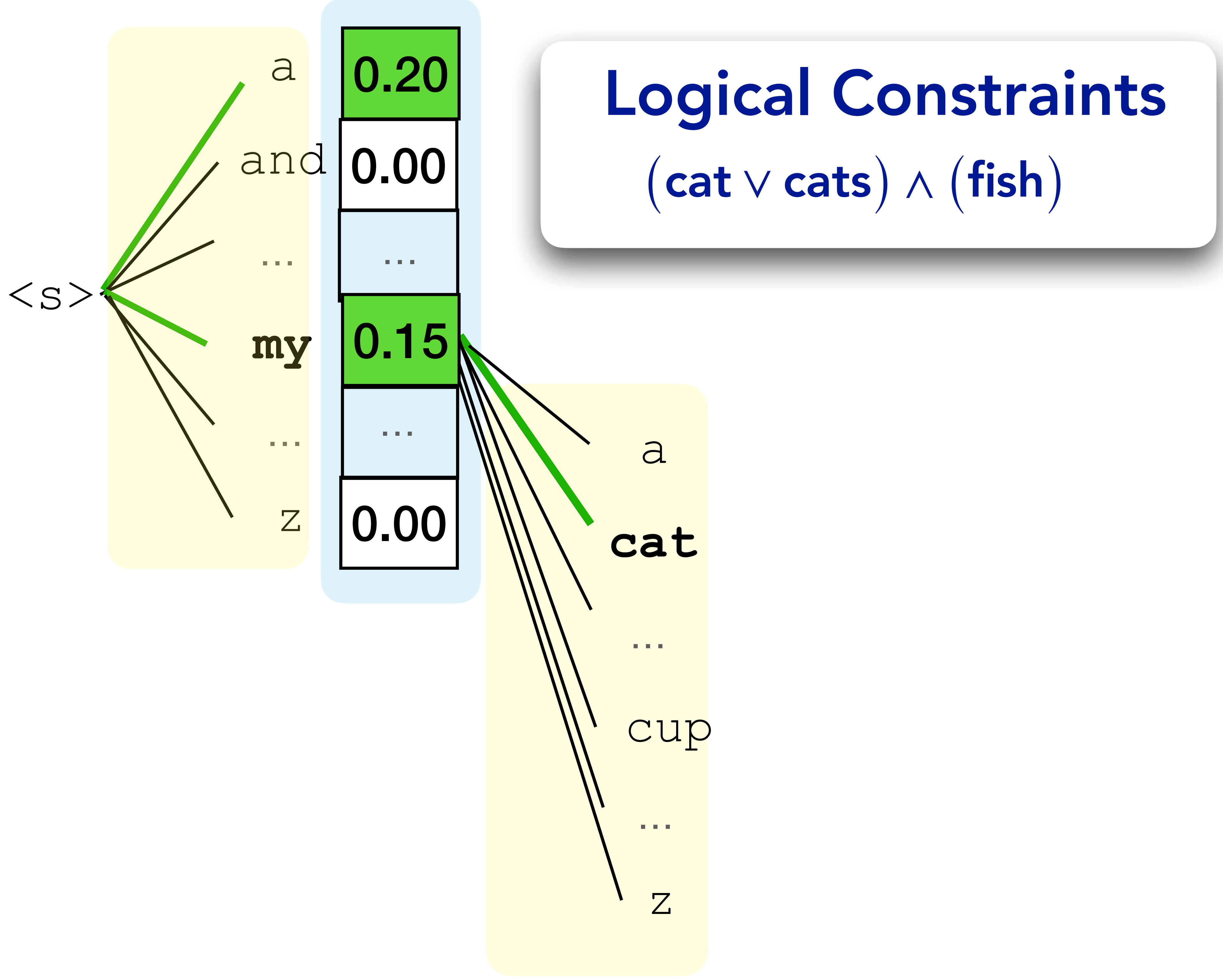
Logical Constraints

(cat ∨ cats) ∧ (fish)

NeuroLogic decoding [Lu et al 2021]

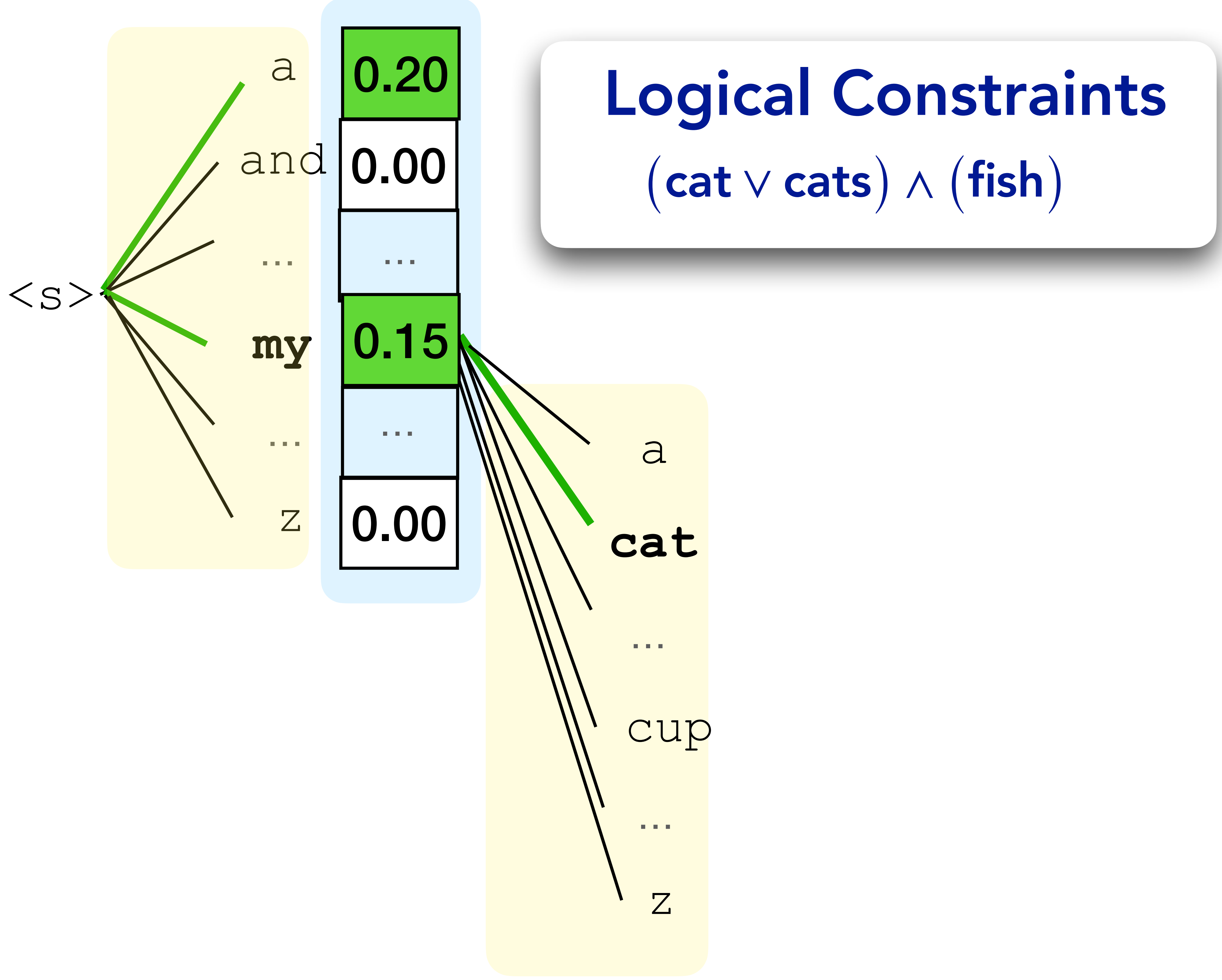
- + favor tokens that [partially] satisfy constraints

- Keep track of remaining constraints



NeuroLogic decoding [Lu et al 2021]

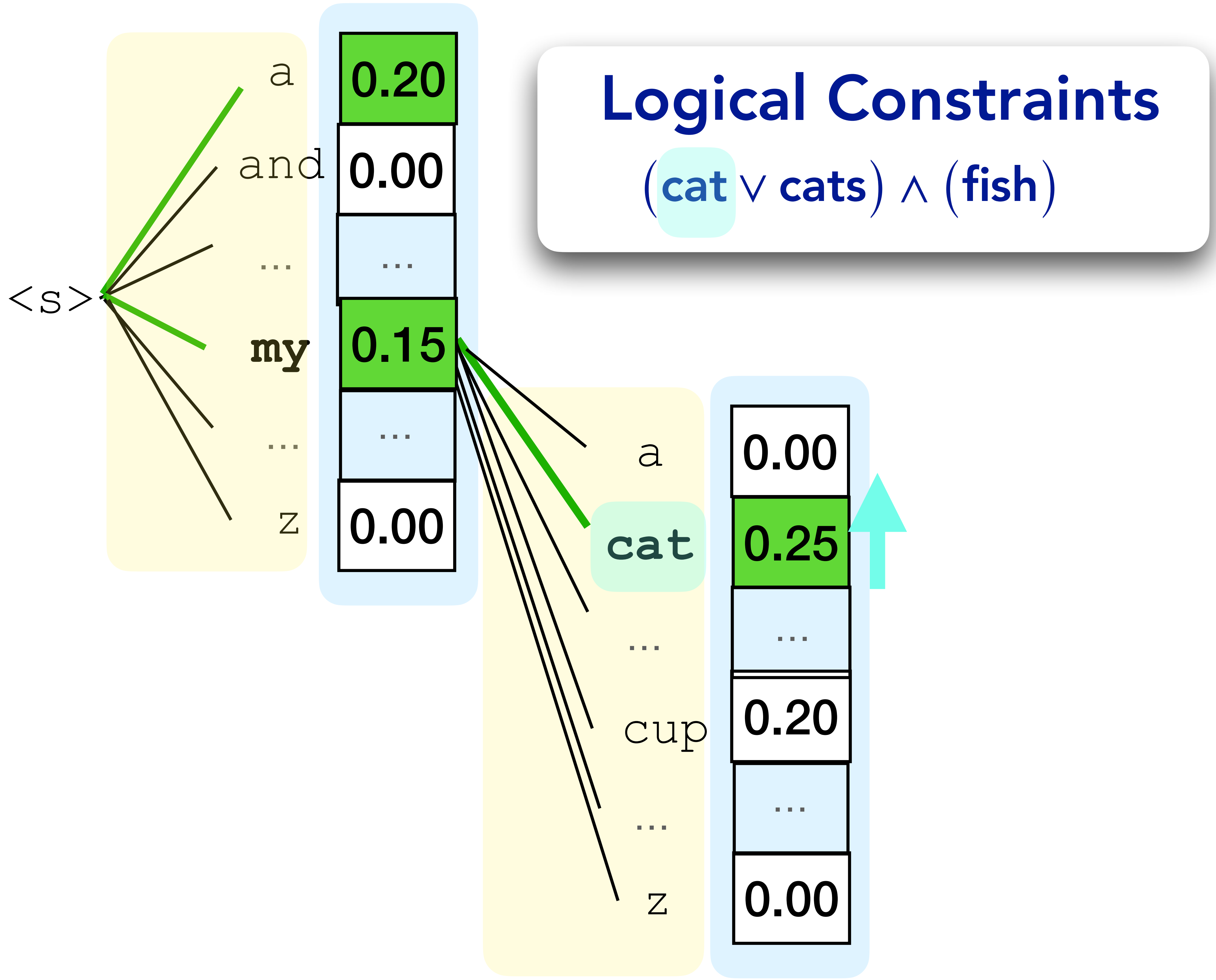
- + favor tokens that [partially] satisfy constraints
- Keep track of remaining constraints



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

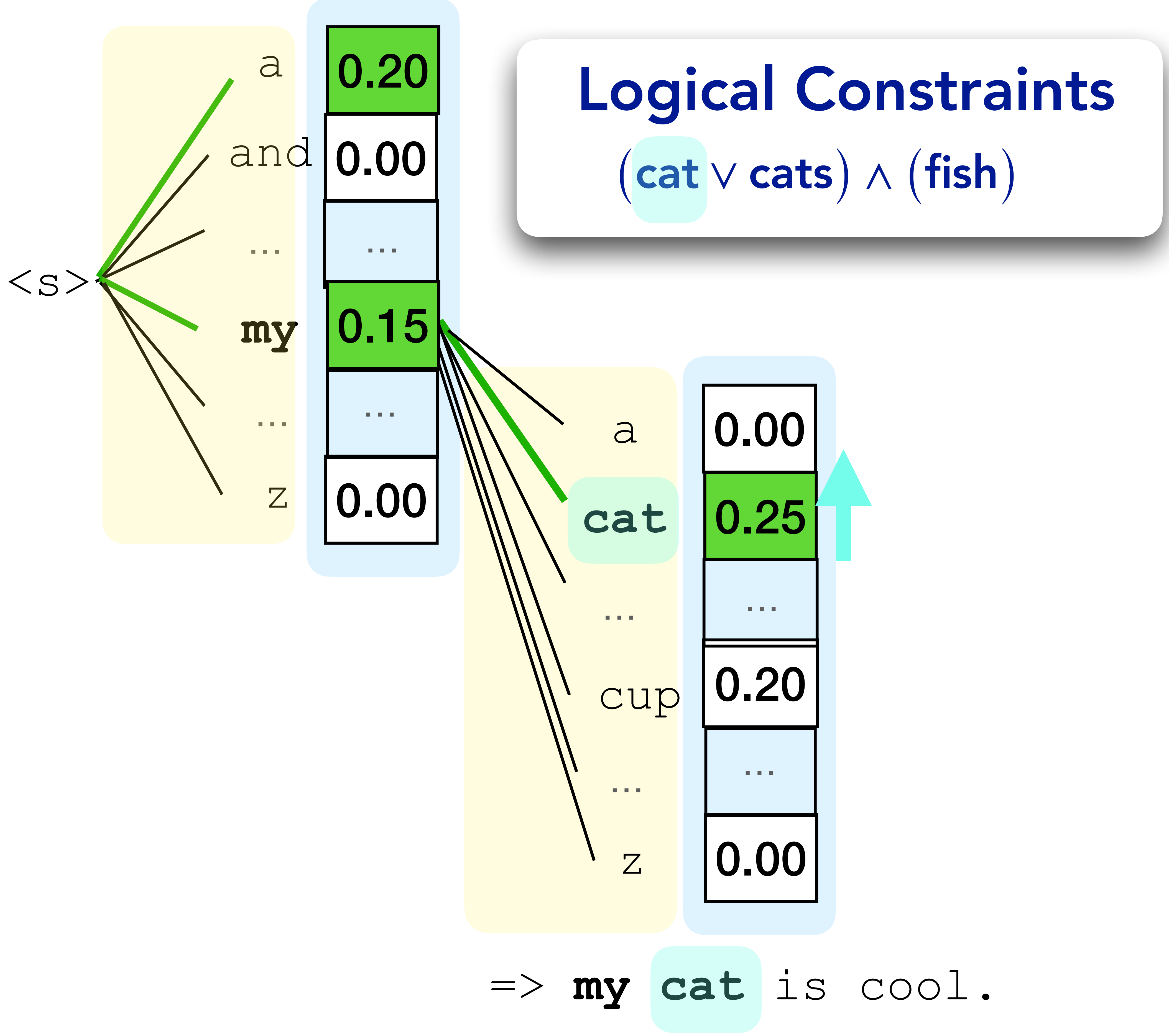


Not trivial!
Details & other features
out of scope for this talk

NeuroLogic decoding [Lu et al 2021]

- + favor tokens that [partially] satisfy constraints
- Keep track of remaining constraints
  - Score next-tokens using
    - $$\underbrace{\log p_\theta(y_t \mid y_{<t})}_{\text{fluency}} + \underbrace{\lambda \max_{c \in \text{remaining}} c(y_t)}_{\text{constraints}}$$

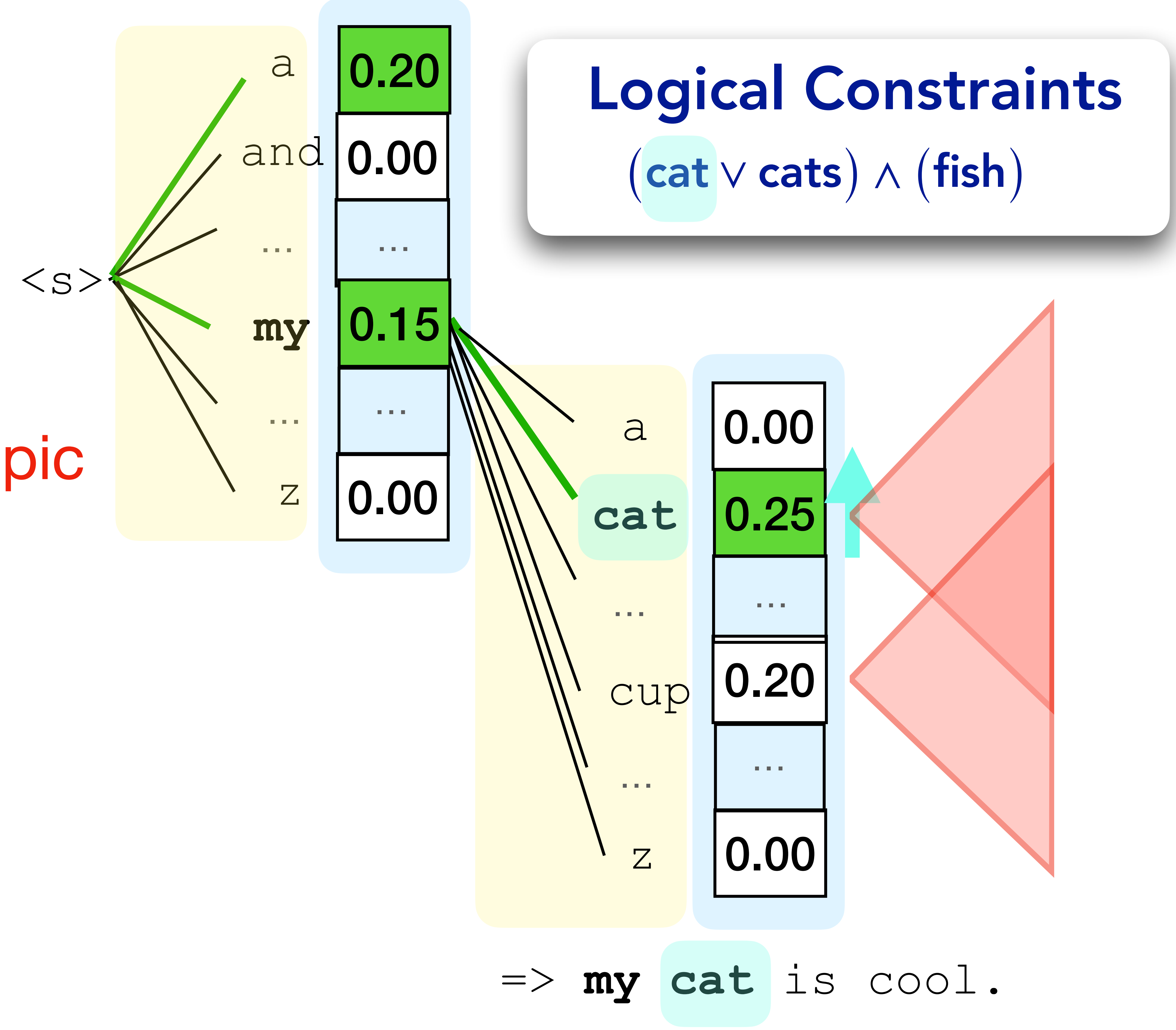
Not trivial!
Details & other features out of scope for this talk



NeuroLogic decoding [Lu et al 2021]

- \+ favor tokens that [partially] satisfy constraints
- Keep track of remaining constraints
  - Score next-tokens using
  - $$\underbrace{\log p_\theta(y_t \mid y_{<t})}_{\text{fluency}} + \lambda \underbrace{\max_{c \in \text{remaining}} c(y_t)}_{\text{constraints}}$$

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

NeuroLogic A*esque decoding

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

Intractable

NeuroLogic A*esque decoding

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

$$\operatorname{argtopk}_{y_t}\left(\underbrace{\max_{\mathbf{y}_{>t}} F(\mathbf{y}_{<t}, y_t, \mathbf{y}_{>t})}_{\text{Intractable}}\right)$$

, F = fluency + constraints

- 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

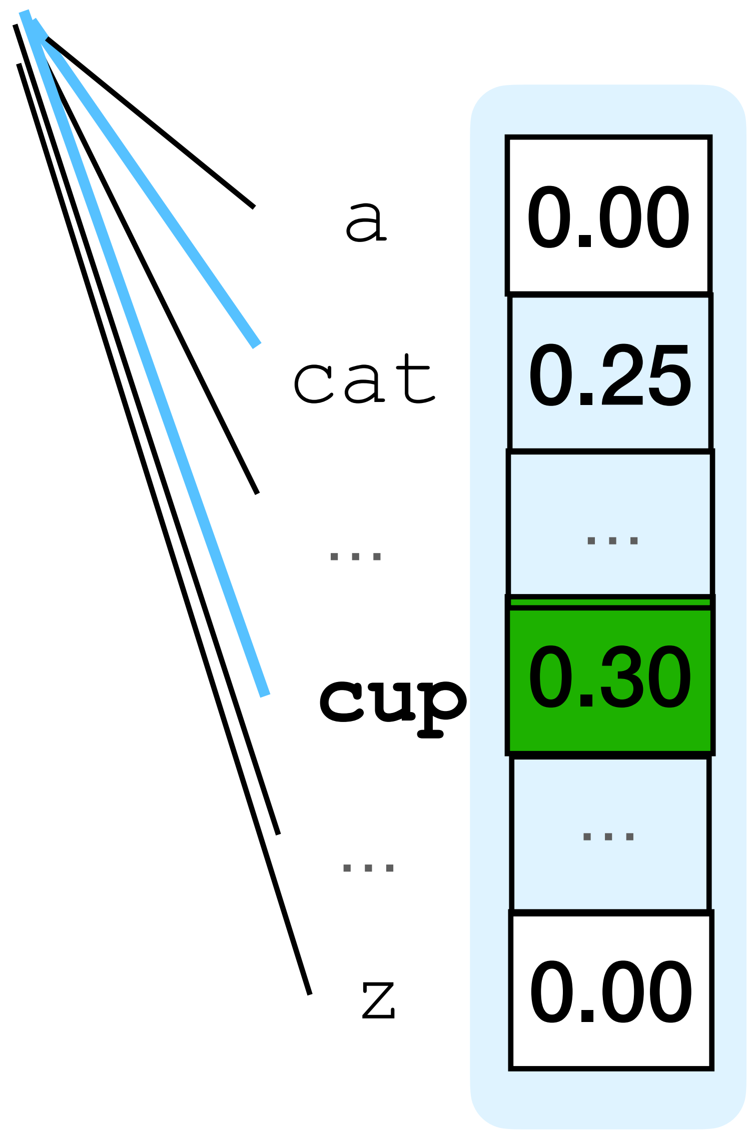
Logical Constraints

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

- Approximate with a *lookahead heuristic*:

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

Fluency + constraints-so-far



NeuroLogic A*esque decoding

Logical Constraints

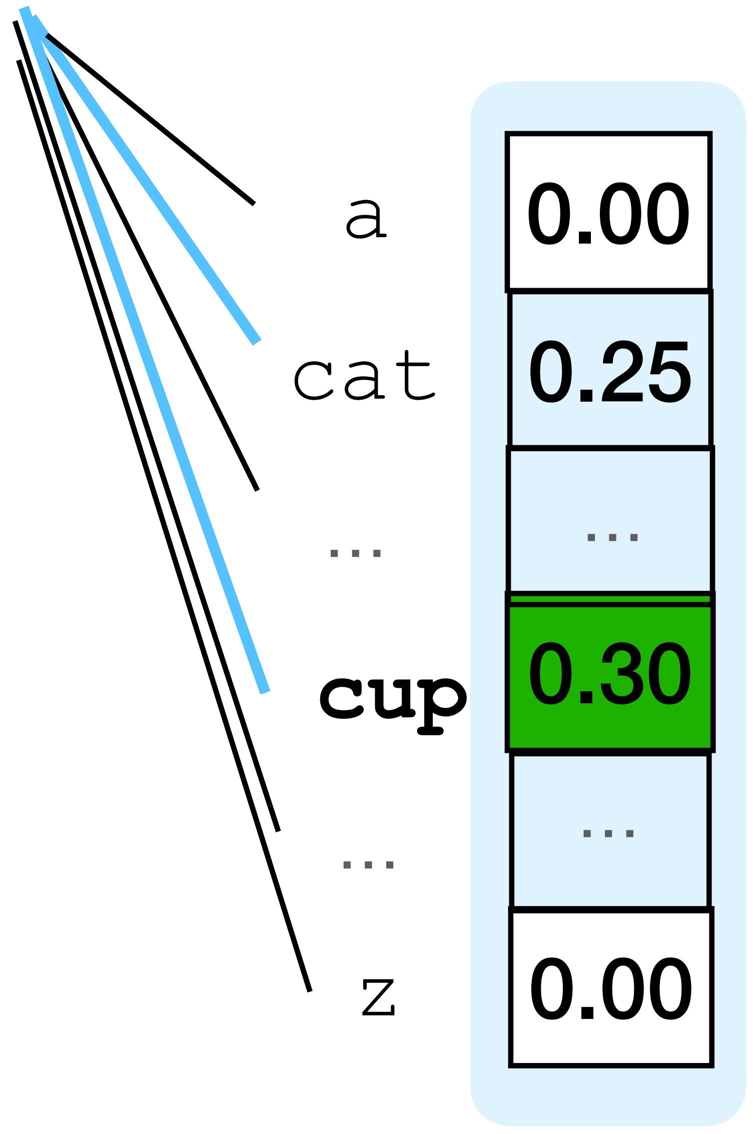
$(\text{cat} \vee \text{cats}) \wedge (\text{fish})$

- Approximate with a *lookahead heuristic*:

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

Fluency + constraints-so-far

Probability of satisfying
Future constraints



NeuroLogic A*esque decoding

Logical Constraints

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

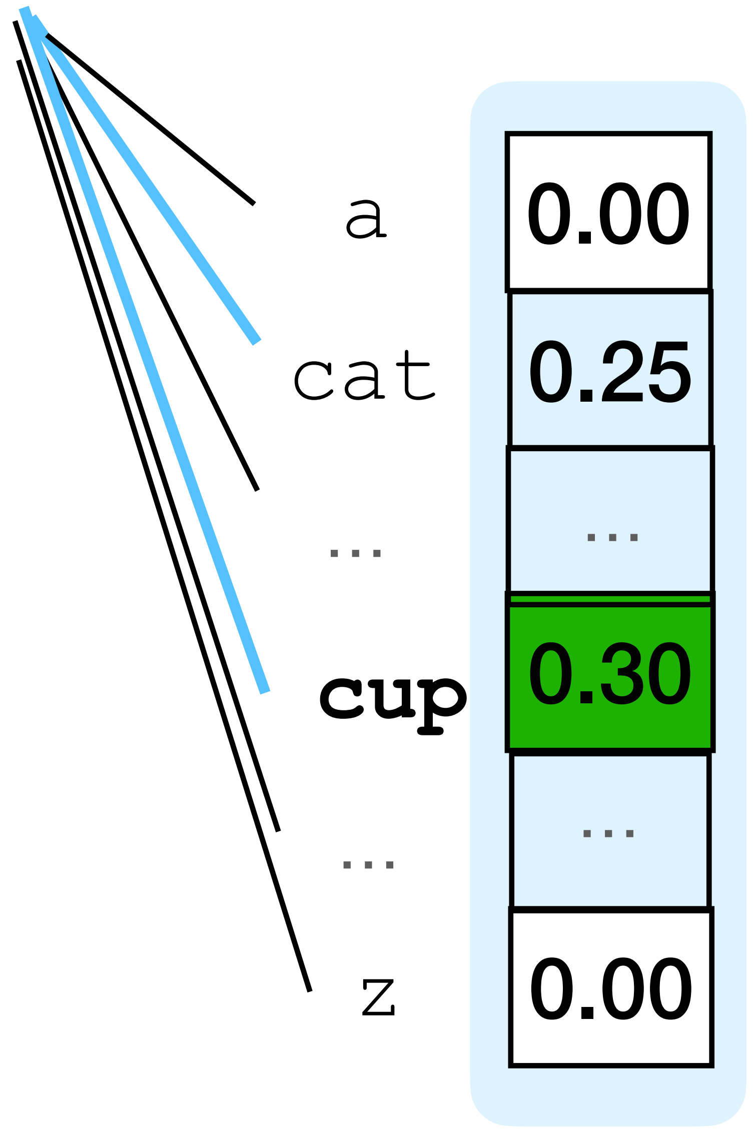
- Approximate with a *lookahead heuristic*:

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

Fluency + constraints-so-far

E.g. single greedy lookahead

Probability of satisfying Future constraints



NeuroLogic A*esque decoding

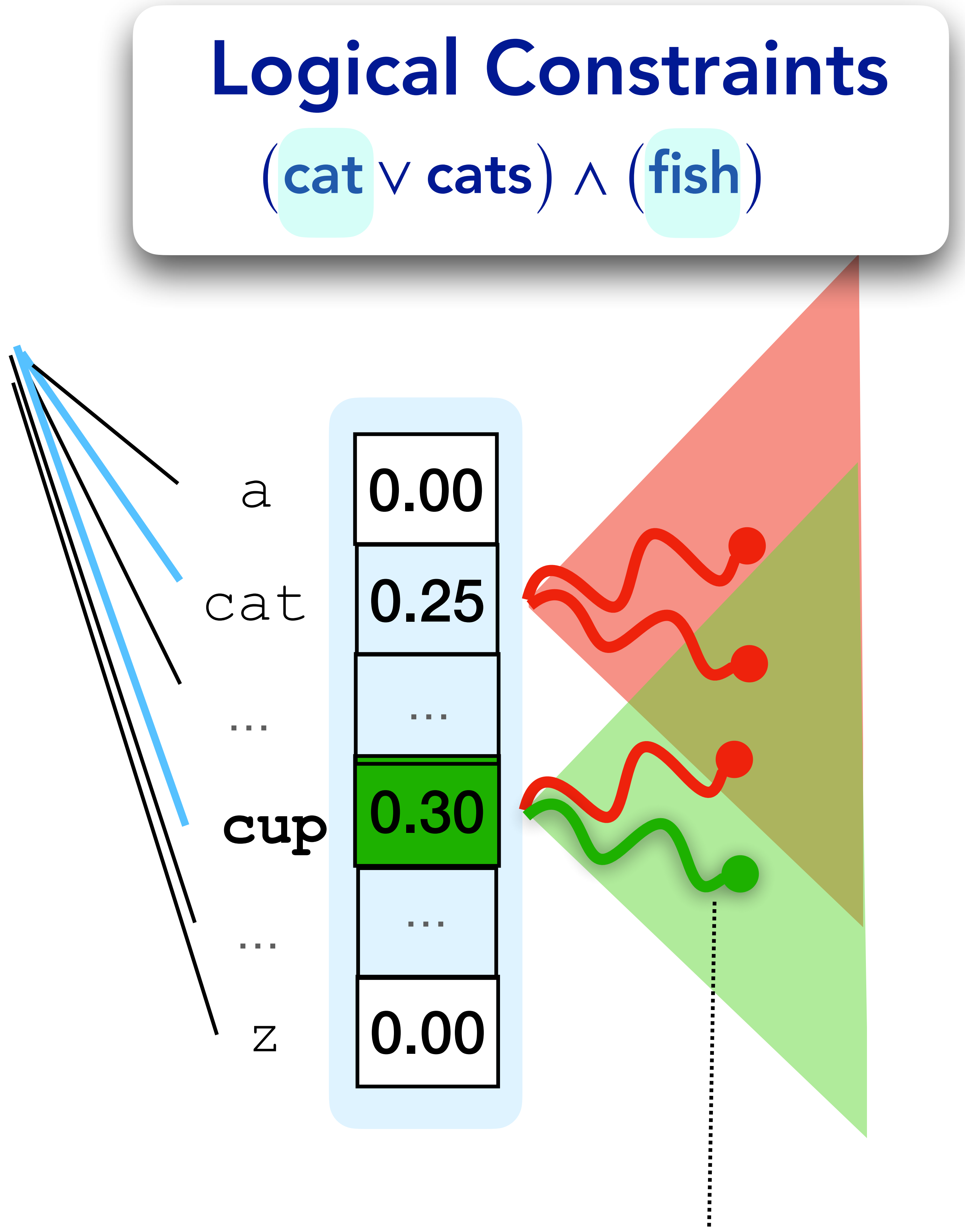
- Approximate with a *lookahead heuristic*:

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

Fluency + constraints-so-far

E.g. single greedy lookahead

Probability of satisfying Future constraints



NeuroLogic A*esque decoding

- Approximate with a *lookahead heuristic*:

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

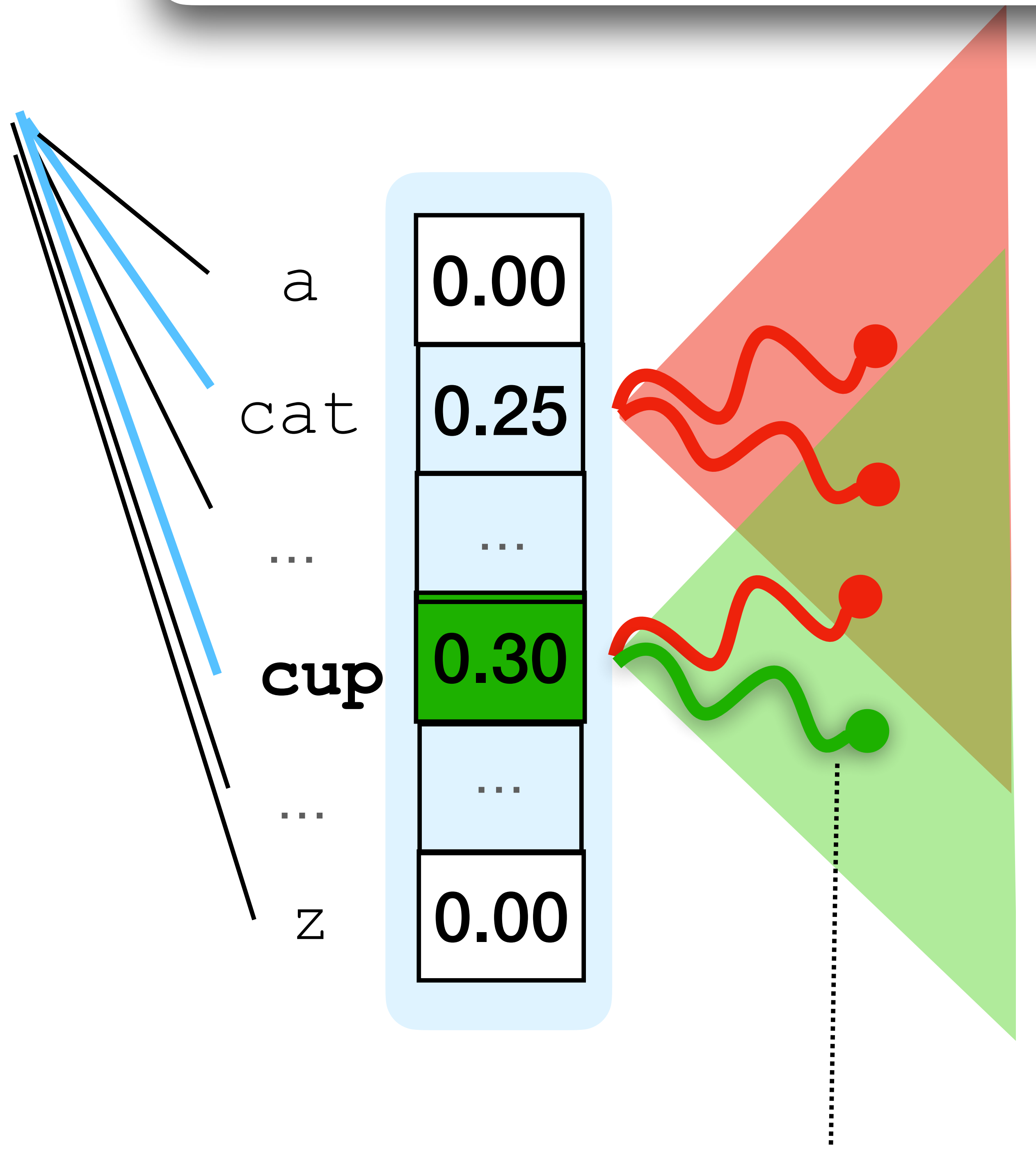
Fluency + constraints-so-far

E.g. single greedy lookahead

Probability of satisfying Future constraints

Logical Constraints

(cat ∨ cats) ∧ (fish)



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

NeuroLogic A*esque decoding

Logical Constraints

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

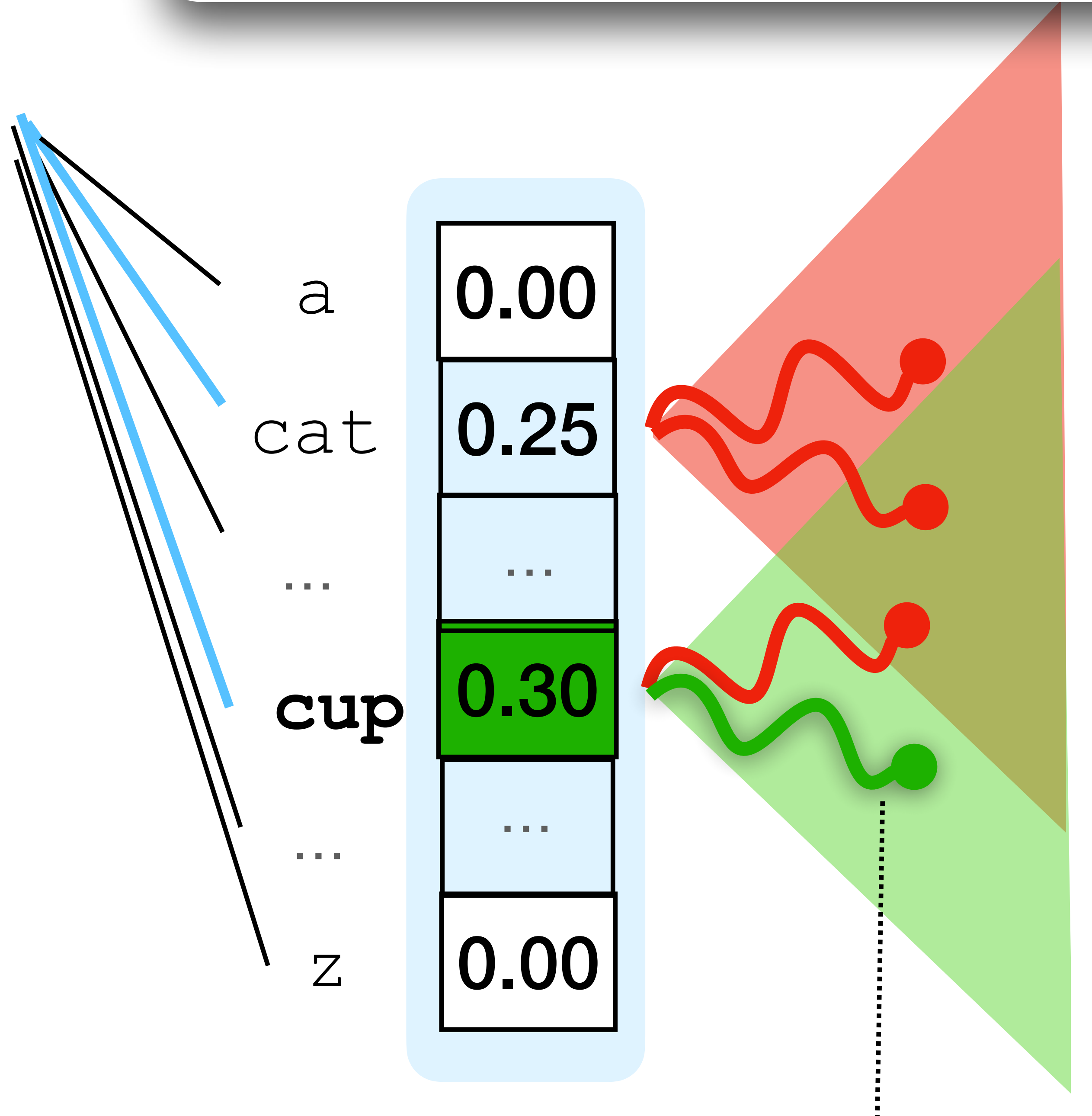
- Approximate with a *lookahead heuristic*:

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

Fluency + constraints-so-far

E.g. single greedy lookahead

Probability of satisfying Future constraints



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

NeuroLogic A*esque decoding

Logical Constraints

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

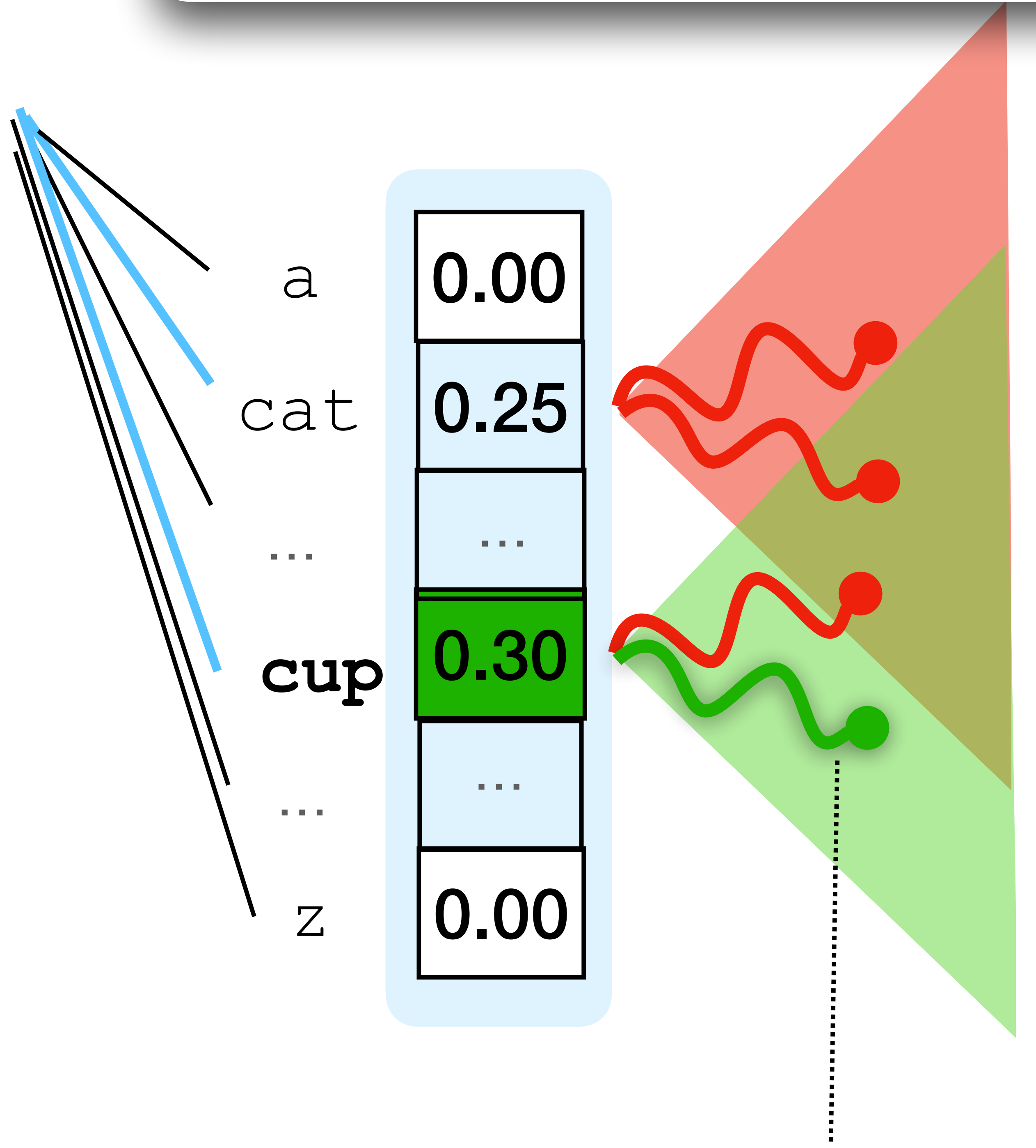
- Approximate with a *lookahead heuristic*:

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

Fluency + constraints-so-far

E.g. single greedy lookahead

Probability of satisfying Future constraints



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

- "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.

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

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

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

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)

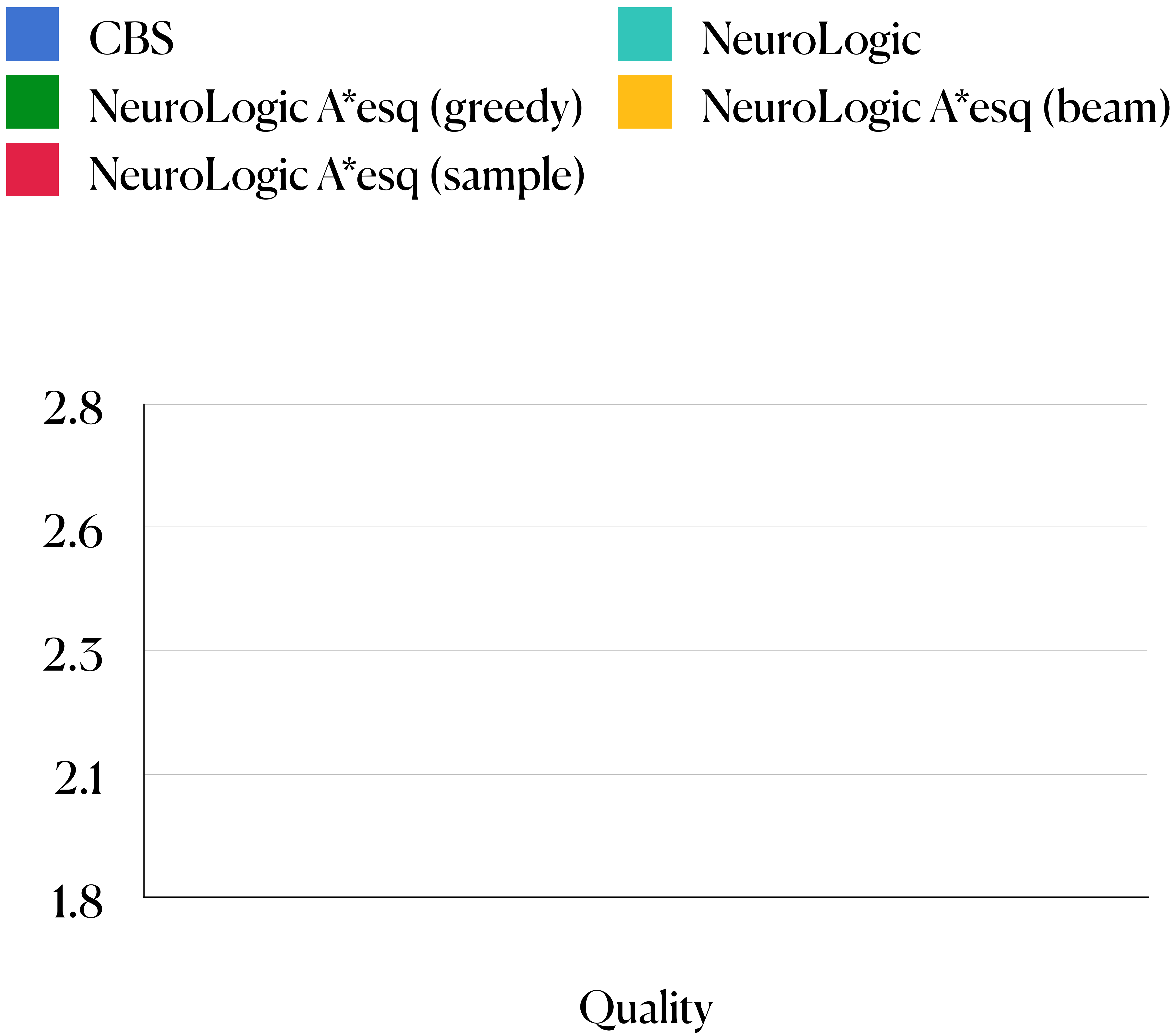
Fine-tuned GPT-2

Off-the-shelf GPT-2

Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

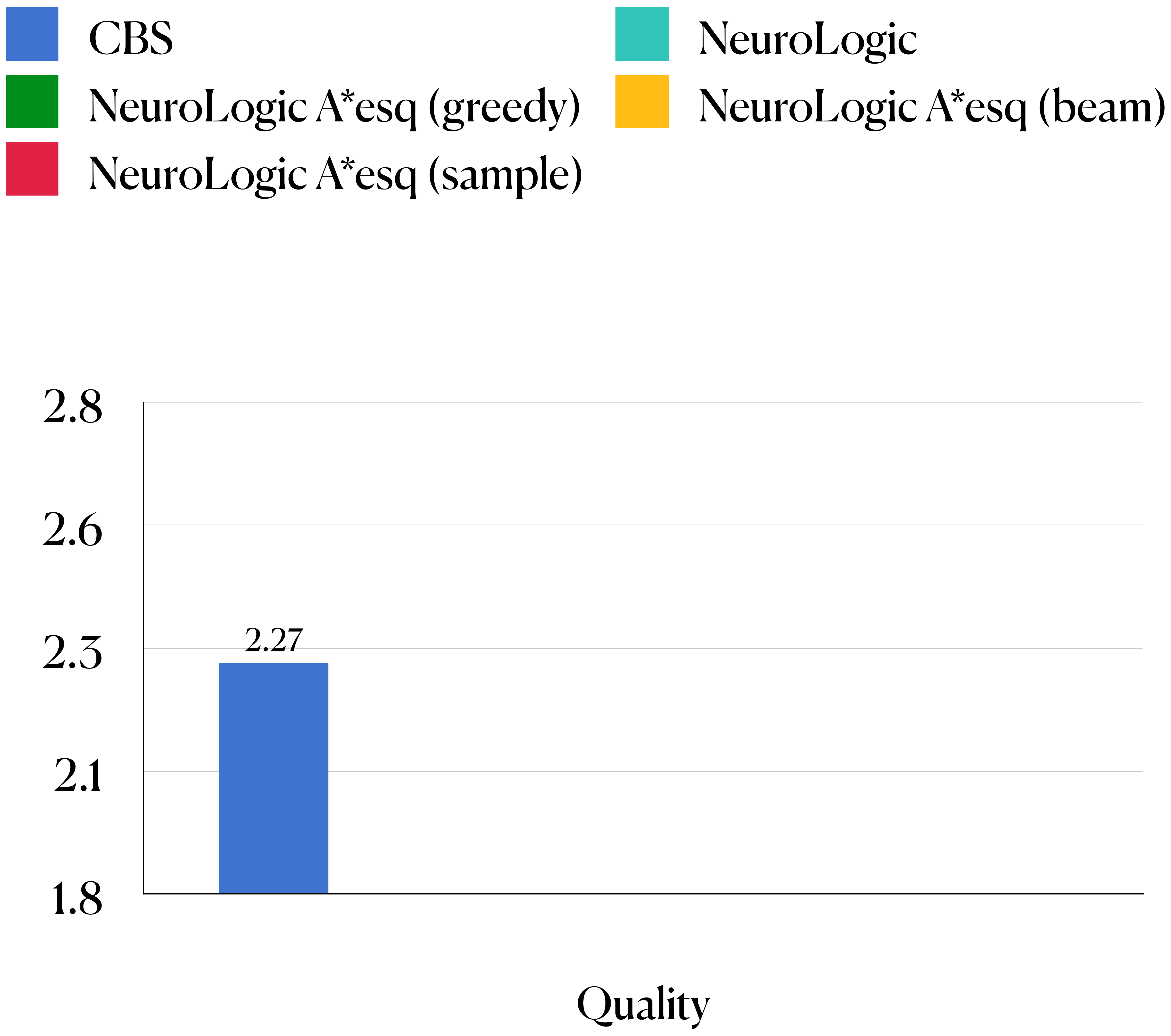


Off-the-shelf GPT-2

Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

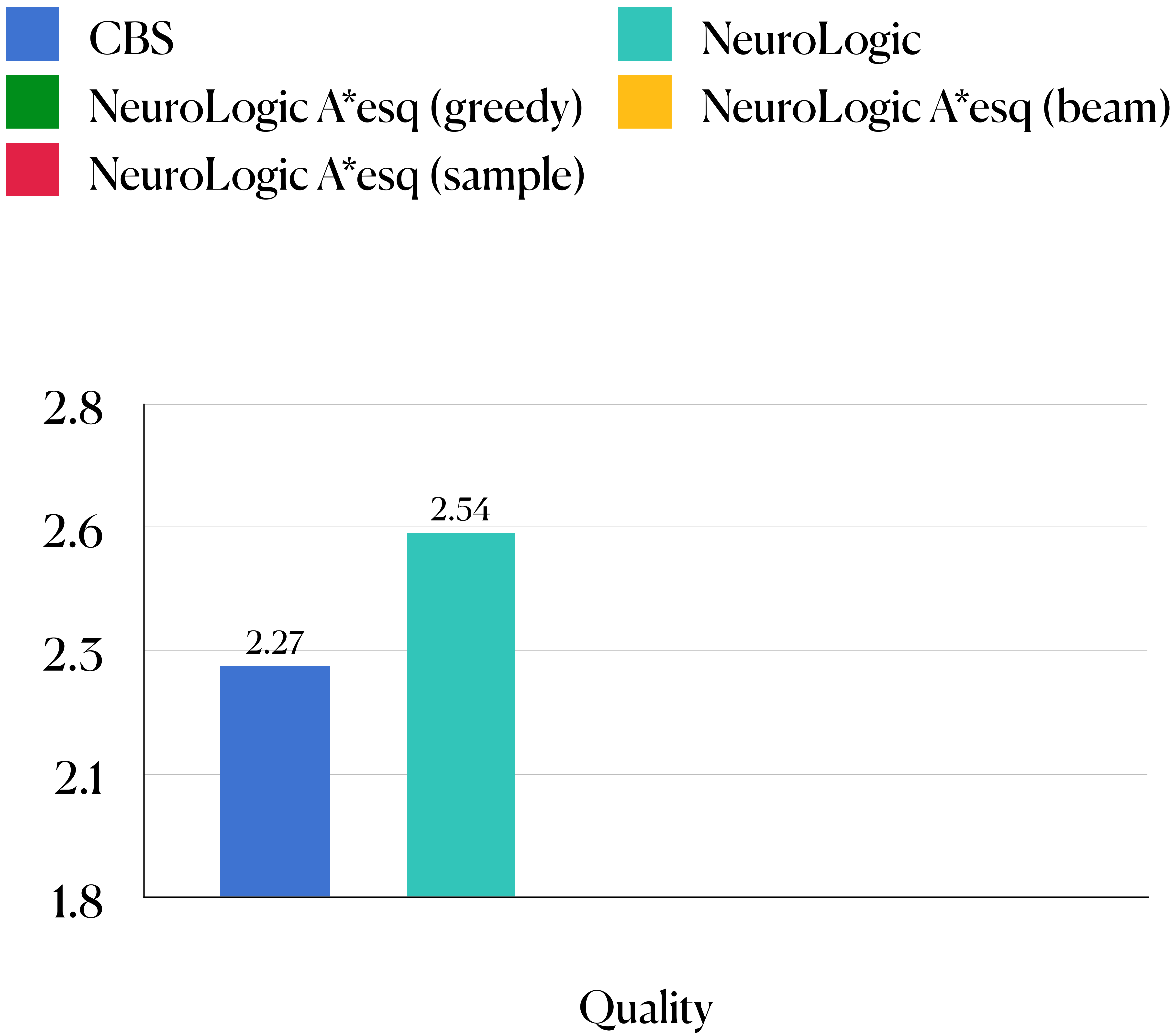


Off-the-shelf GPT-2

Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

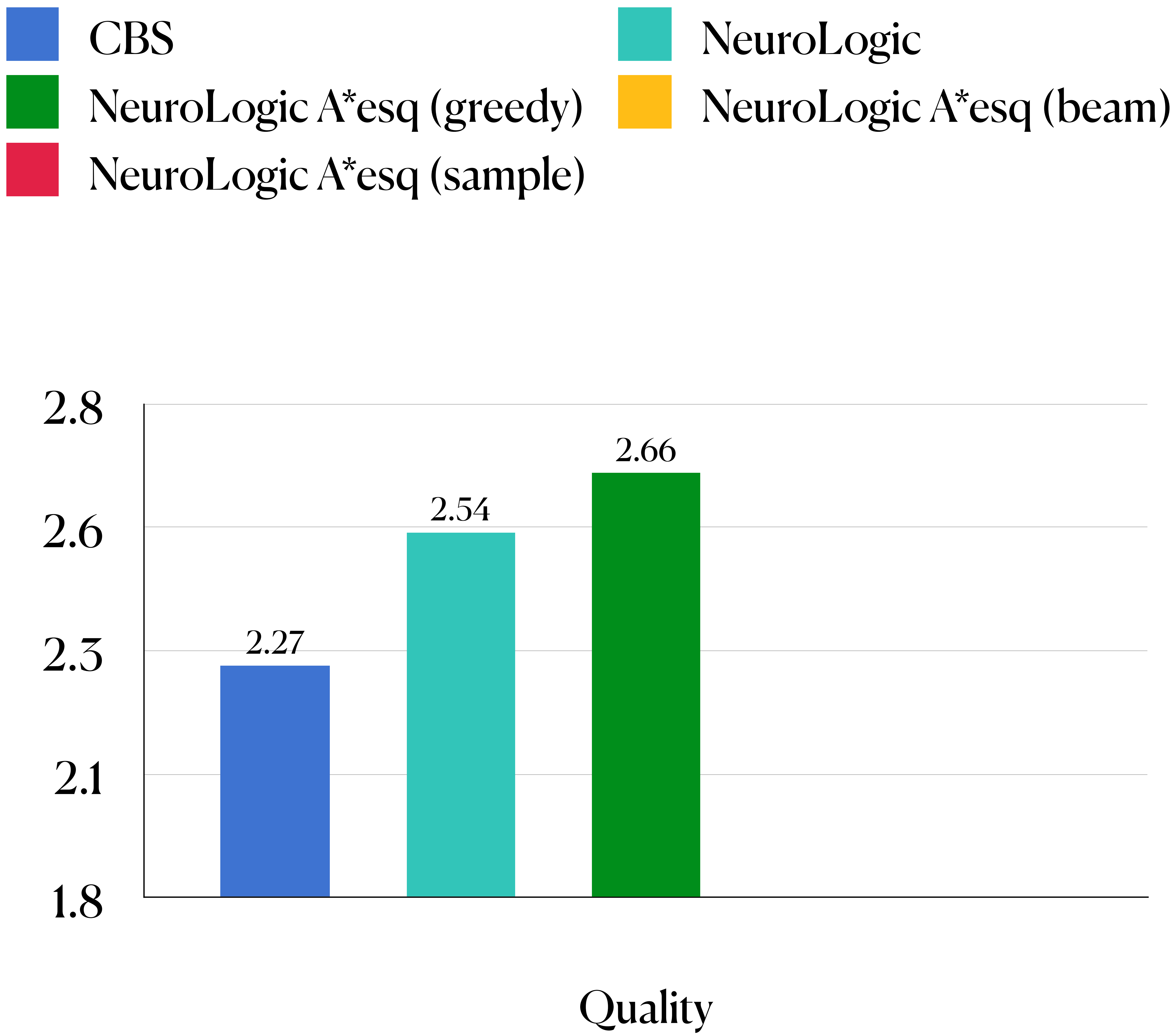


Off-the-shelf GPT-2

Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

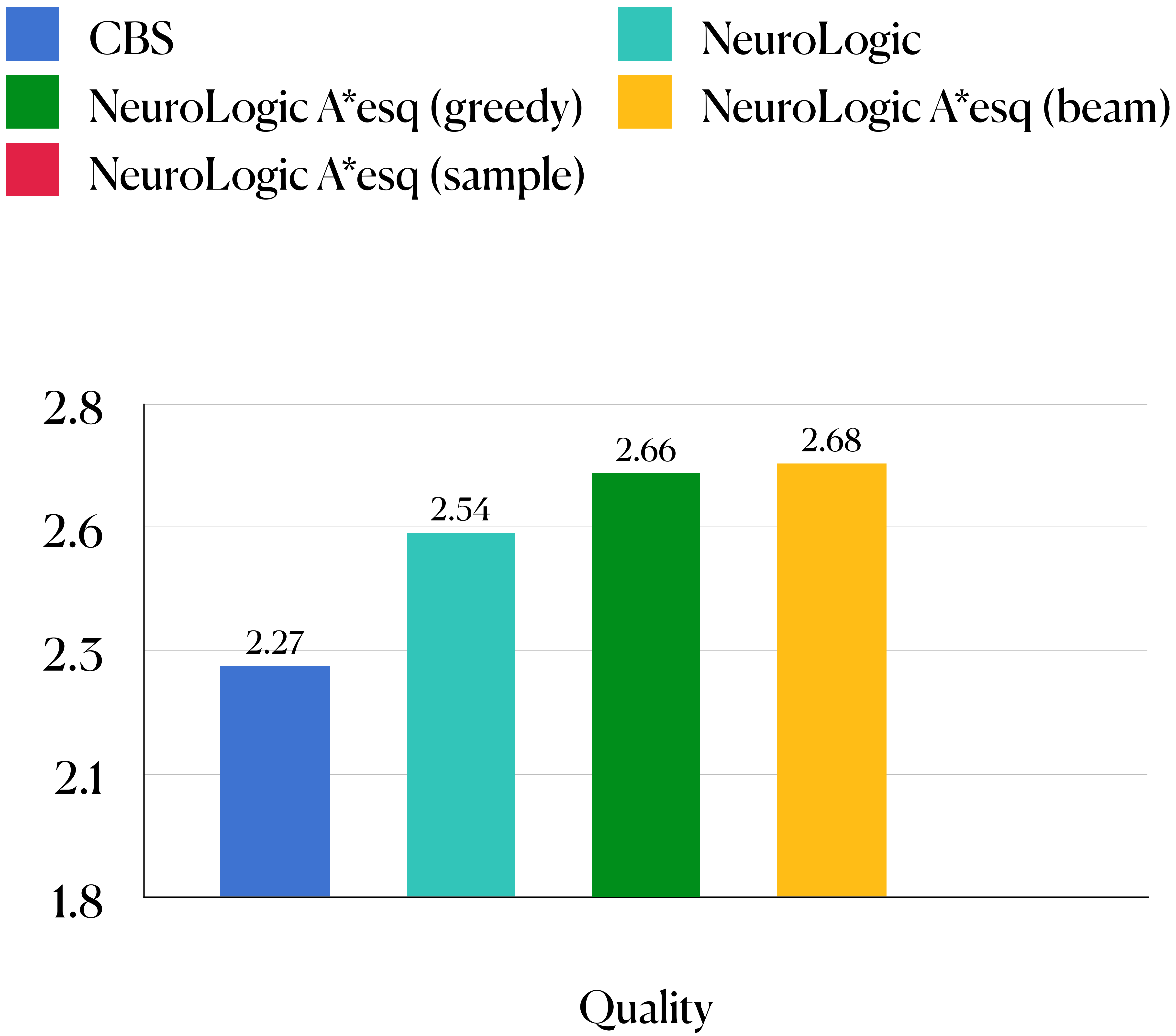


Off-the-shelf GPT-2

Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

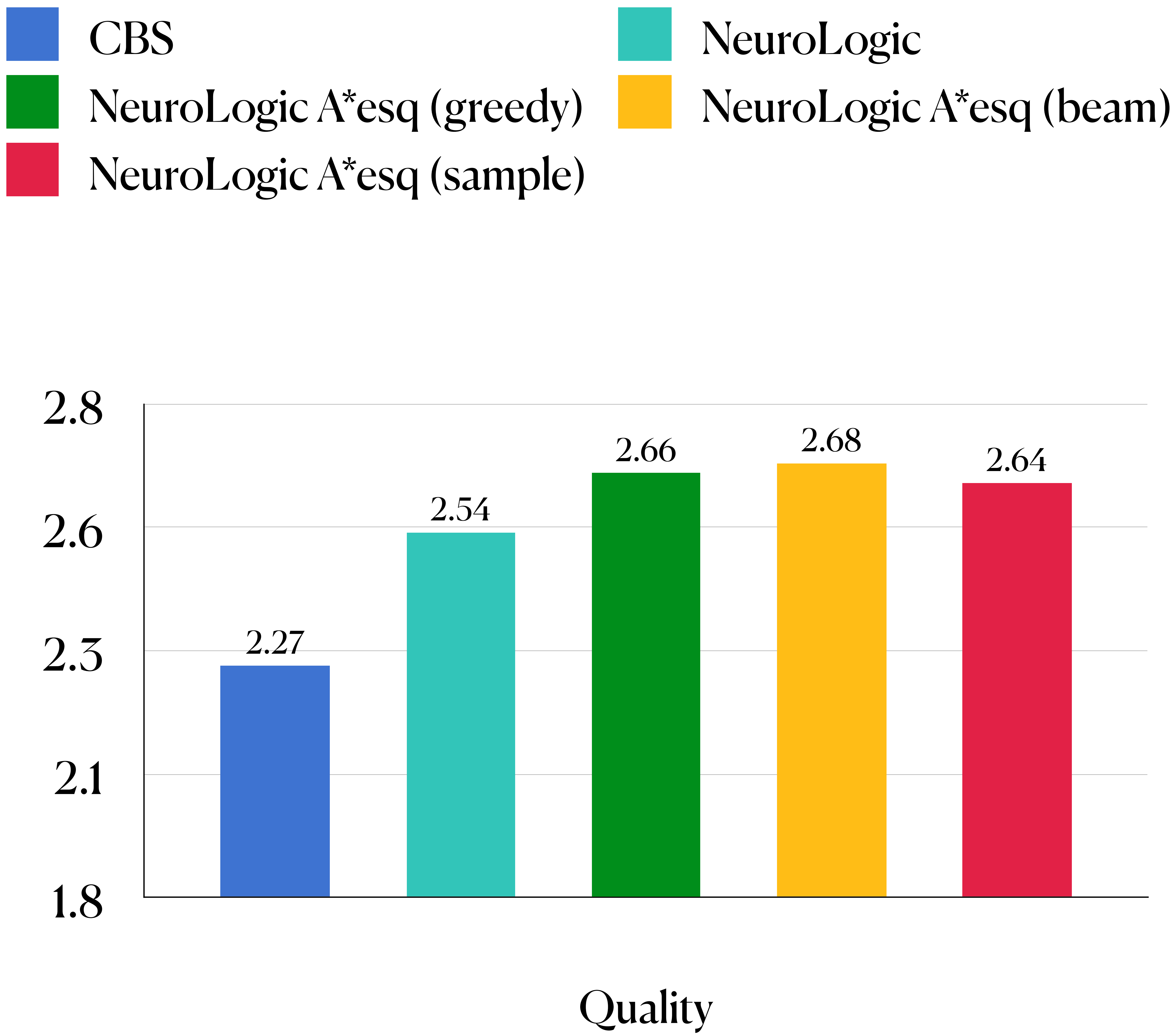


Off-the-shelf GPT-2

Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

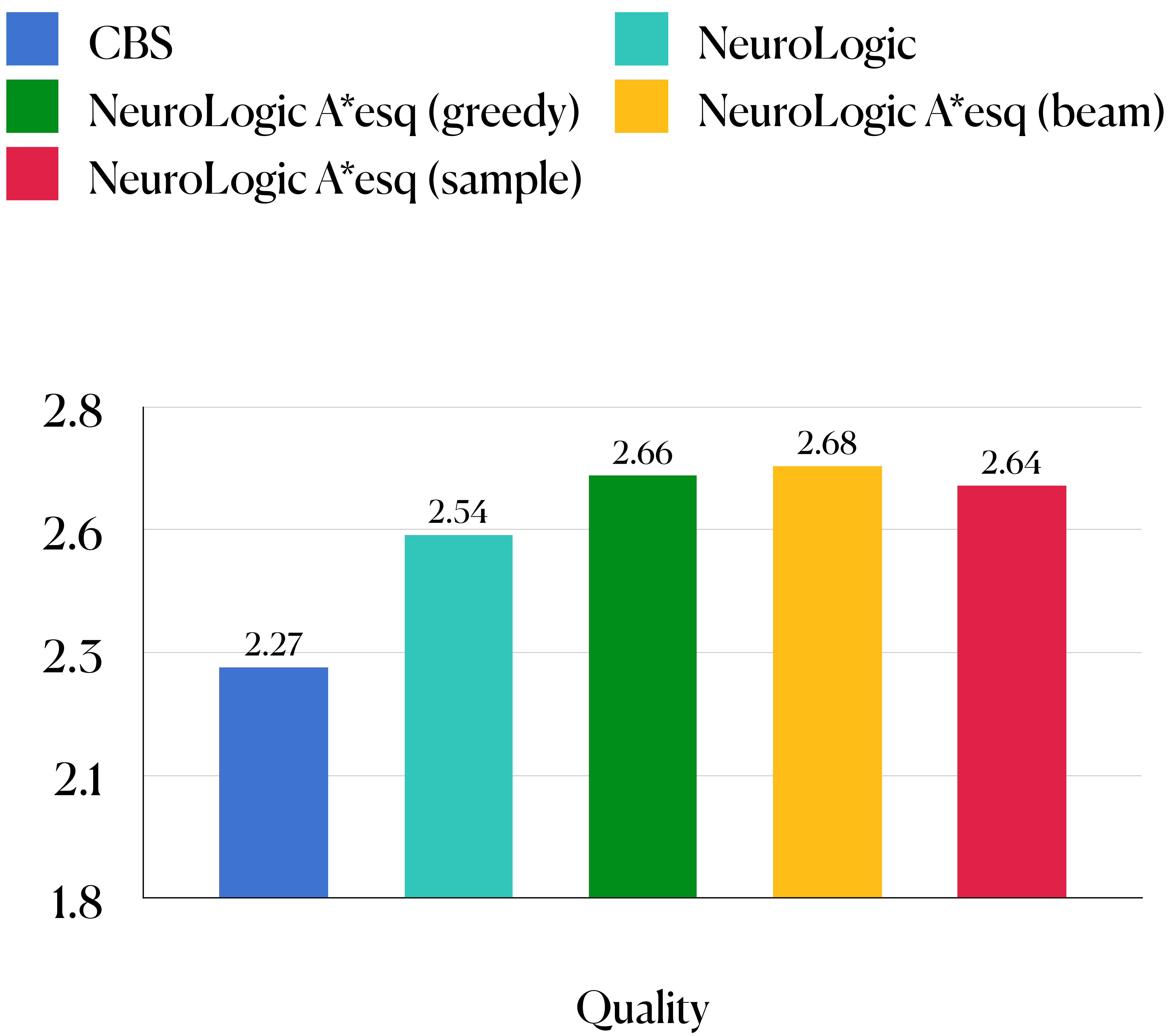


Off-the-shelf GPT-2

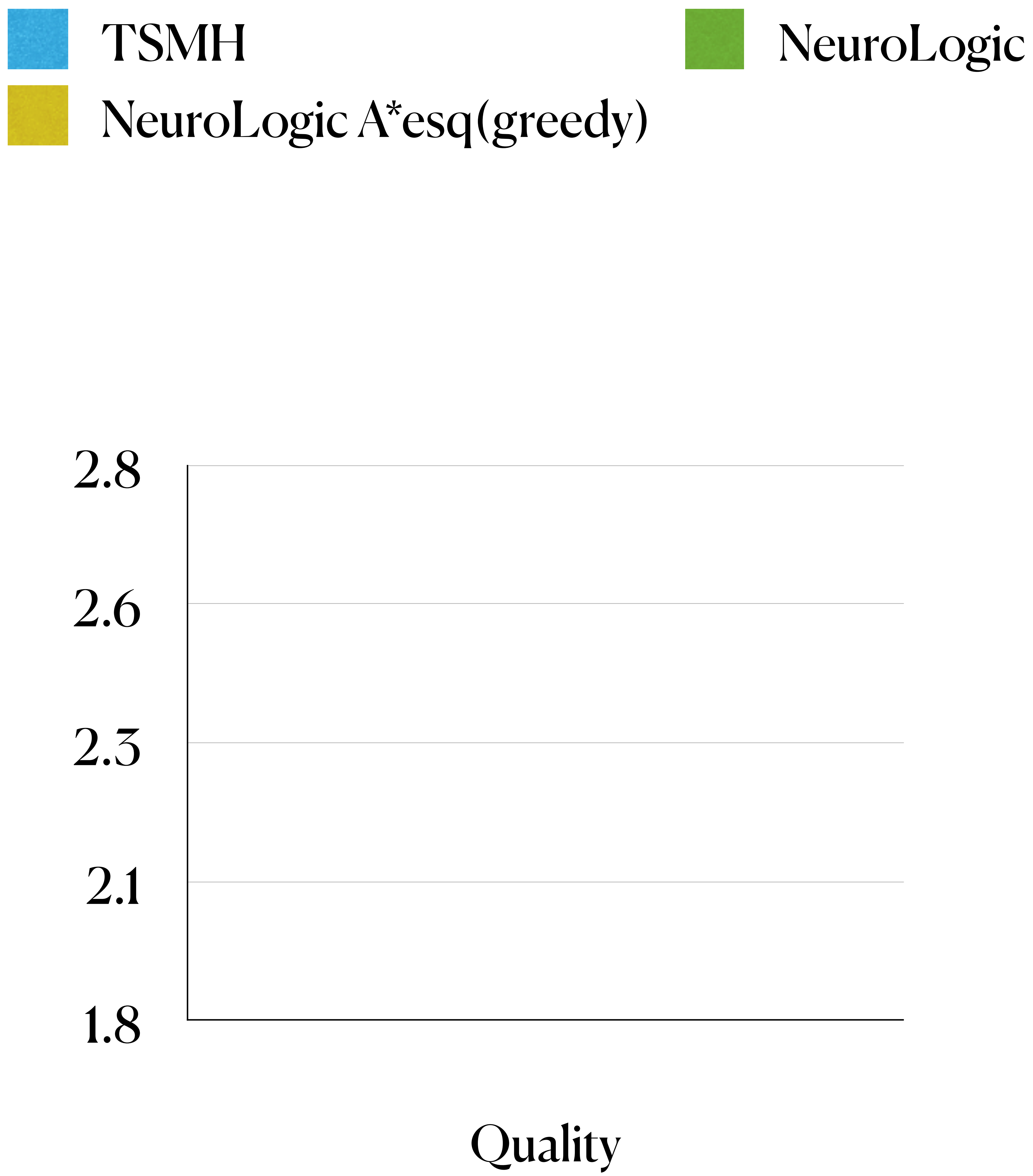
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

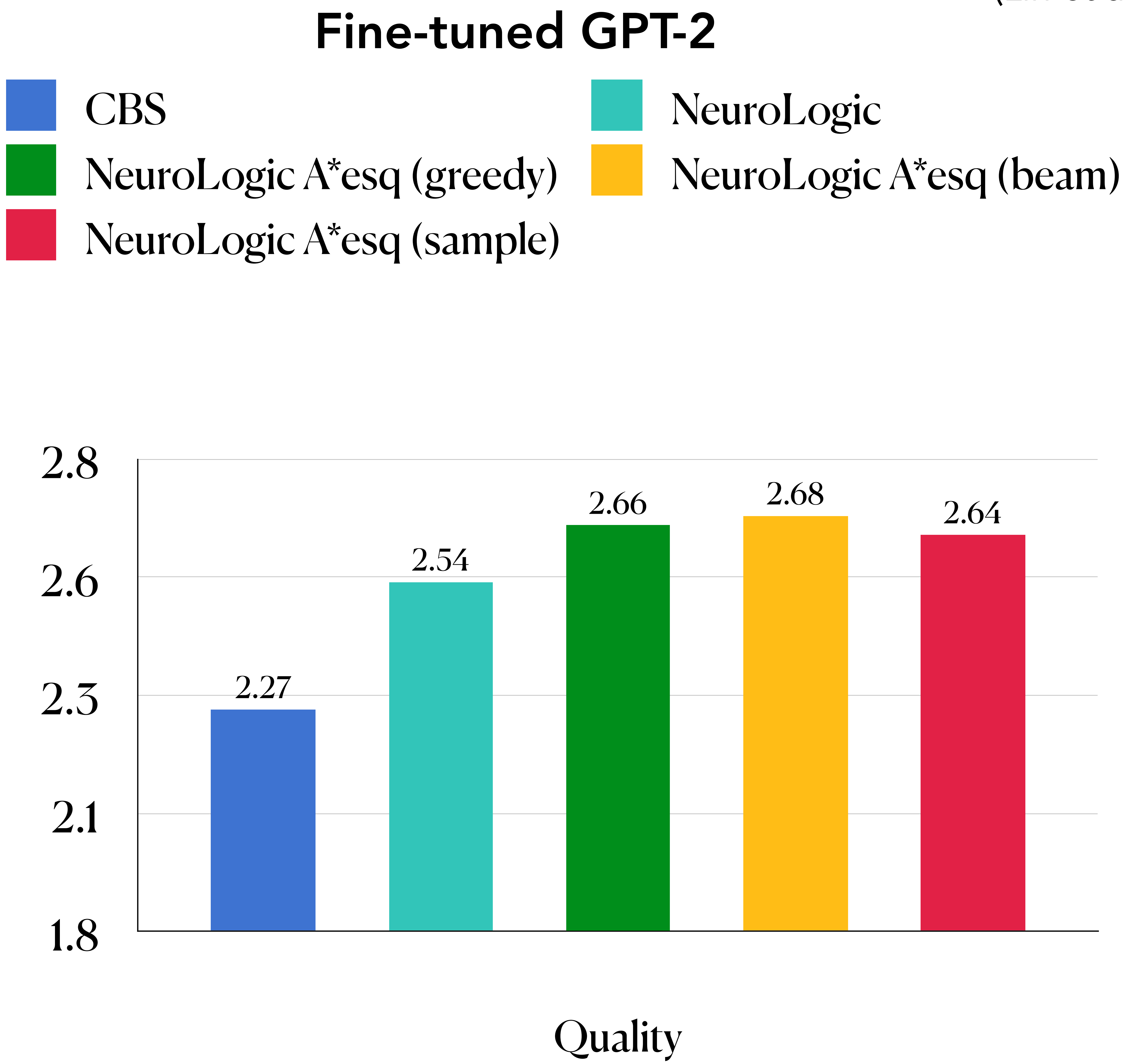


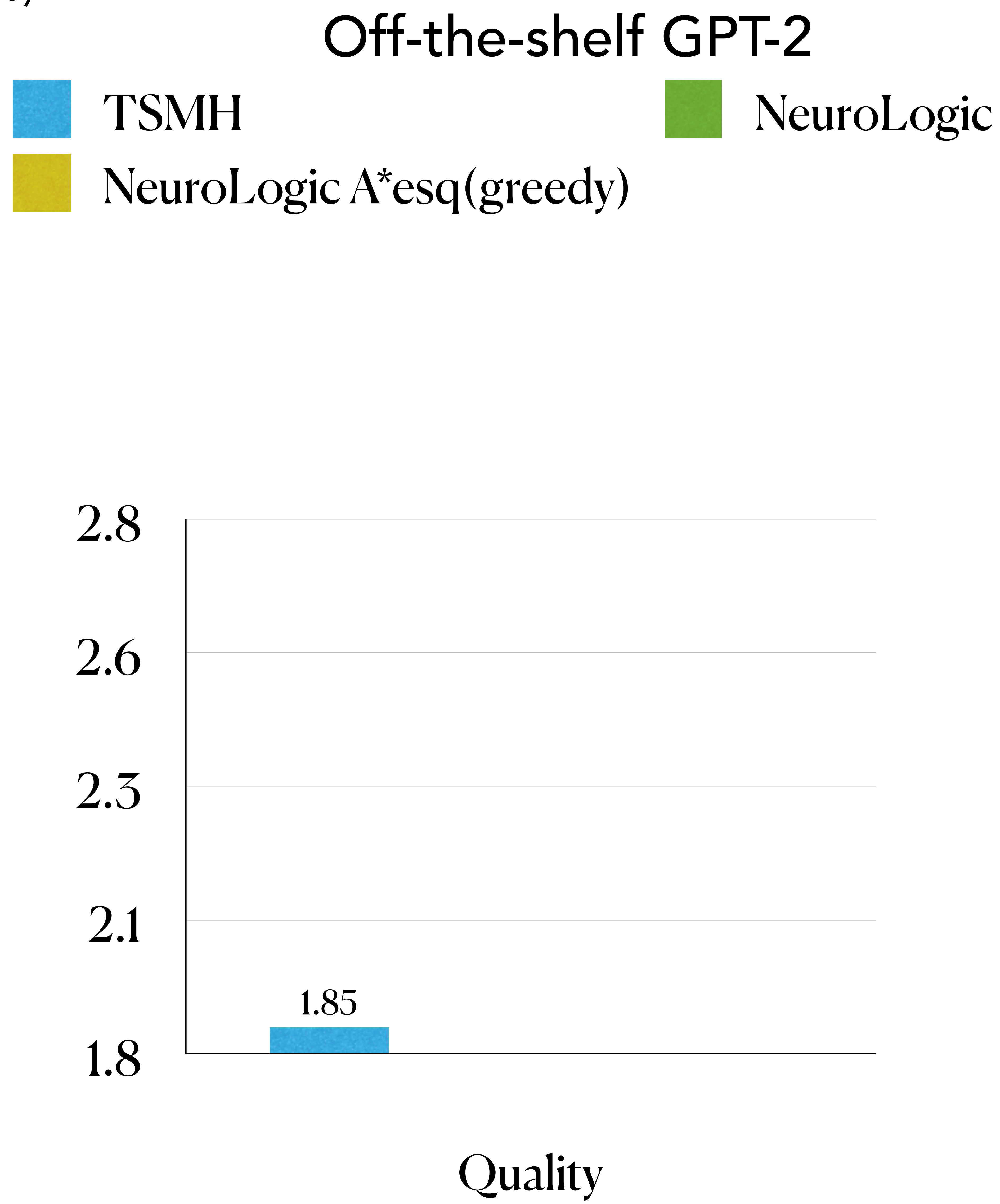
Off-the-shelf GPT-2



Human evaluation | CommonGen

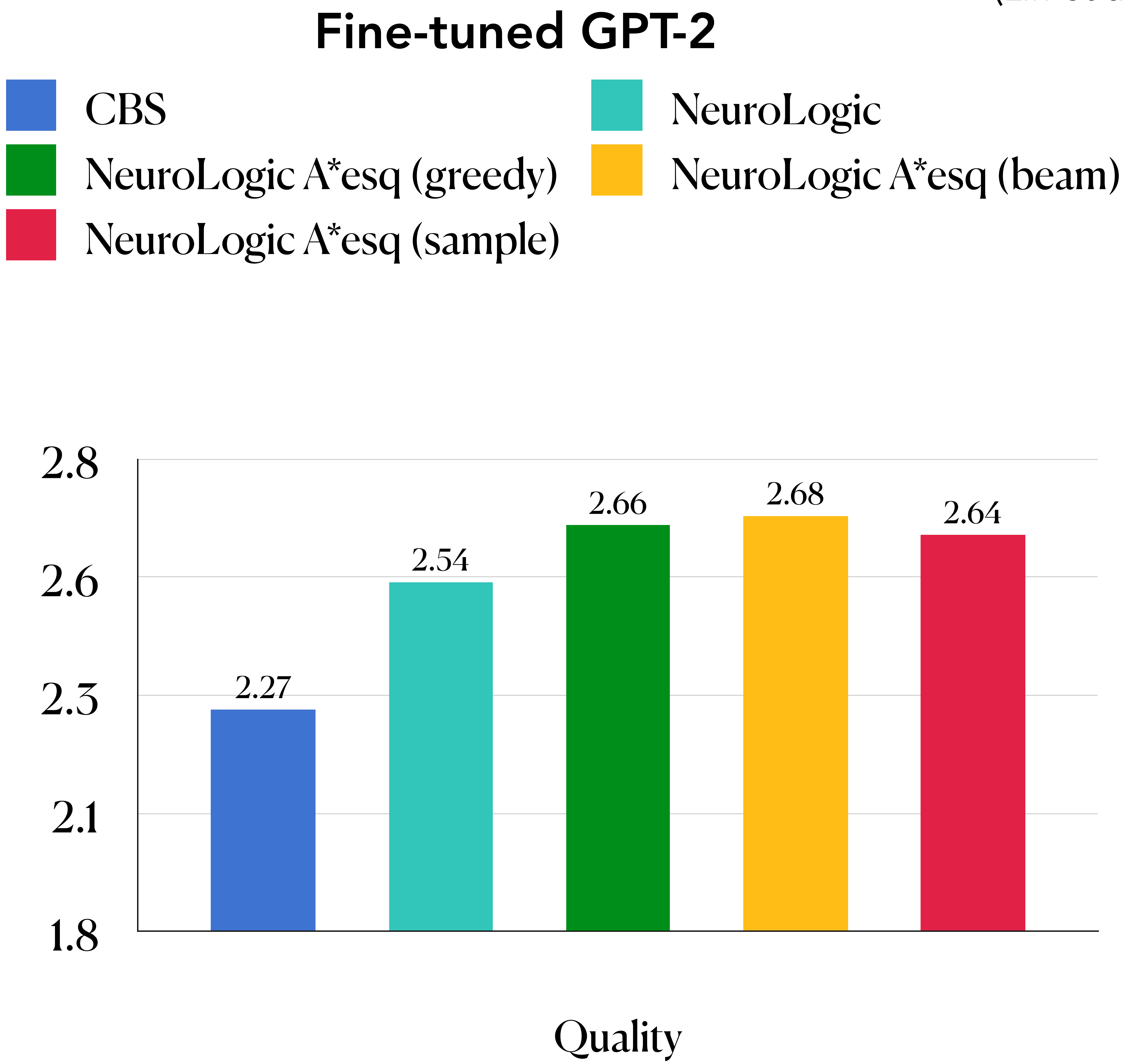
(Lin et al., 2020)

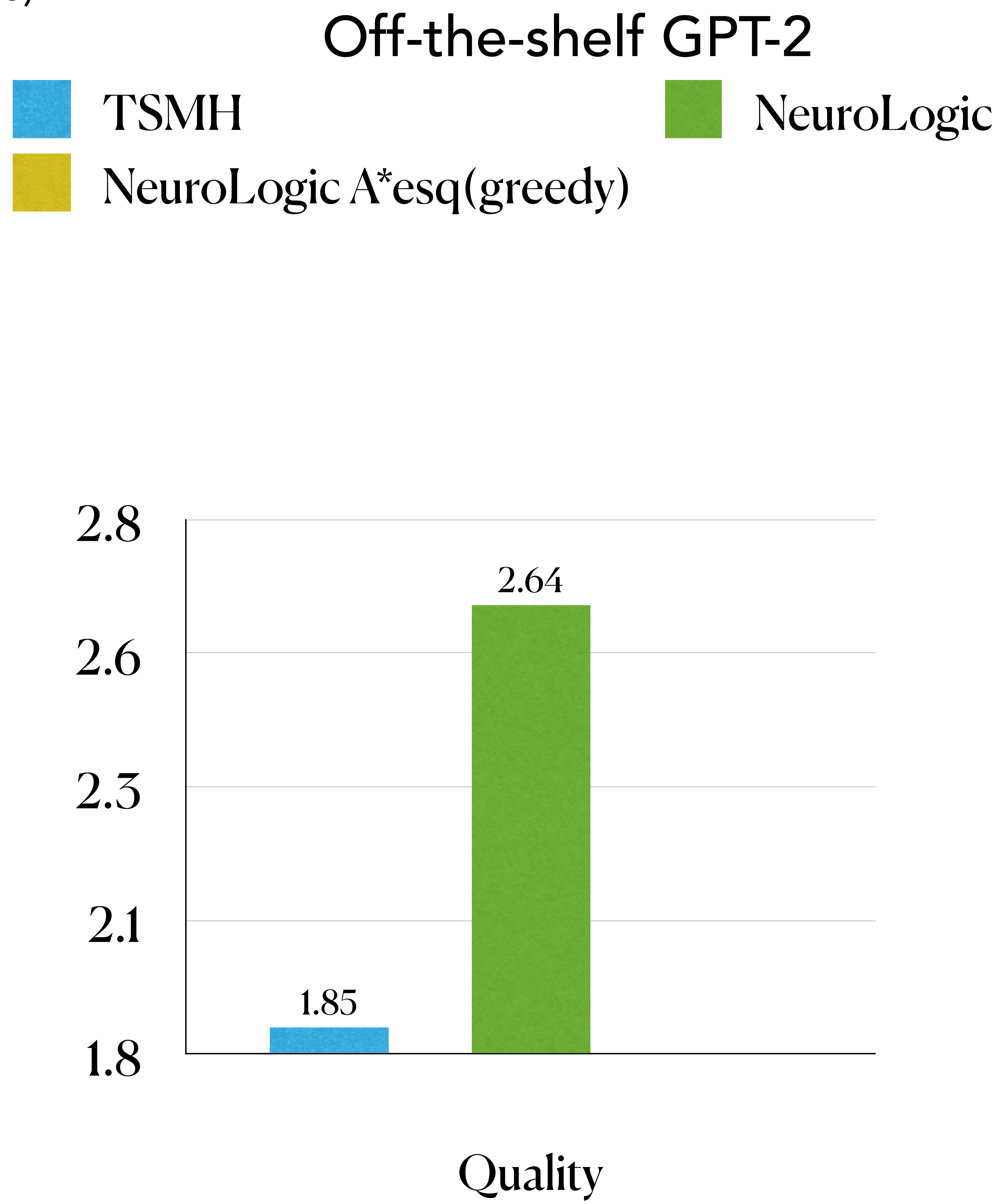




Human evaluation | CommonGen

(Lin et al., 2020)

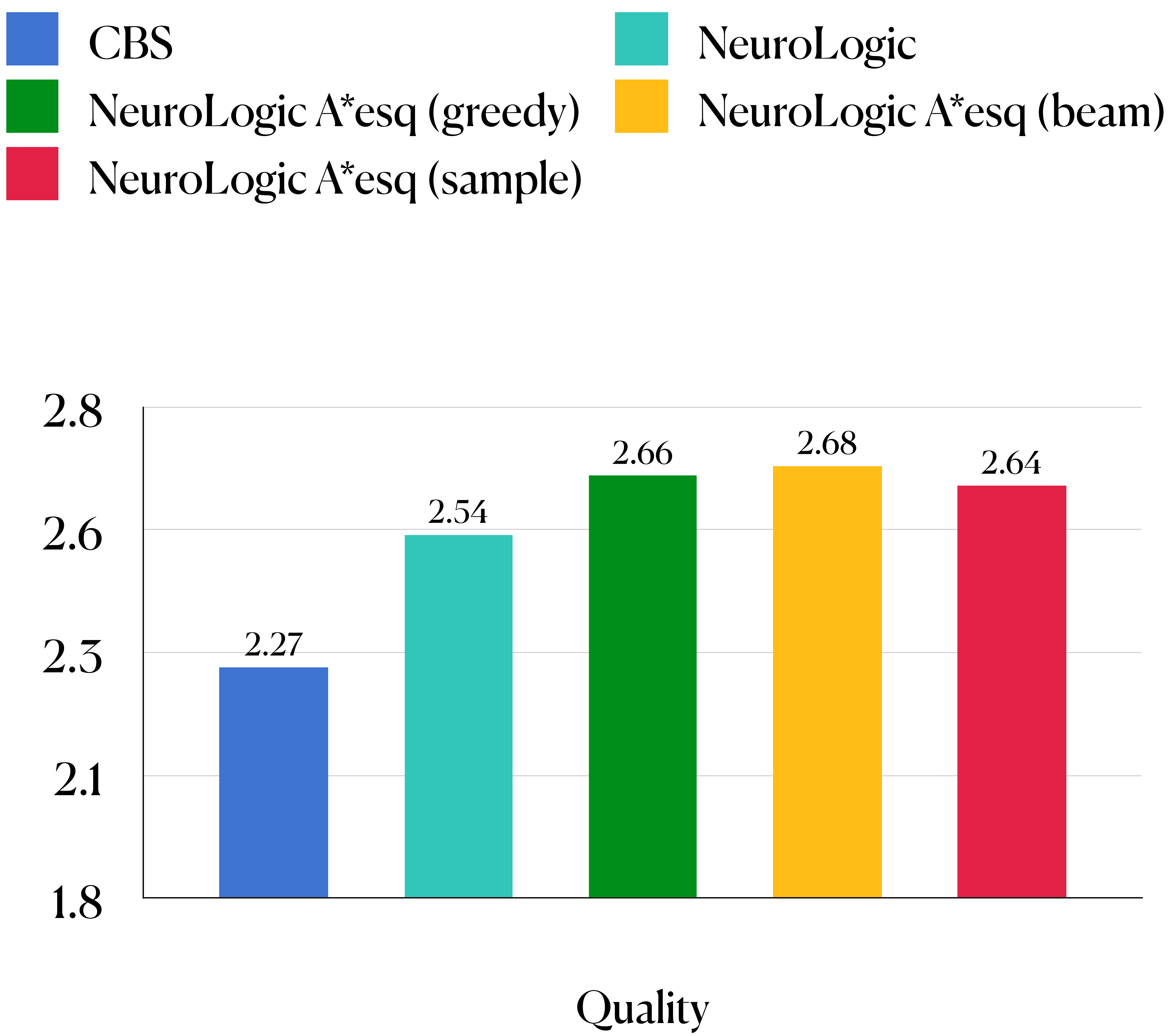




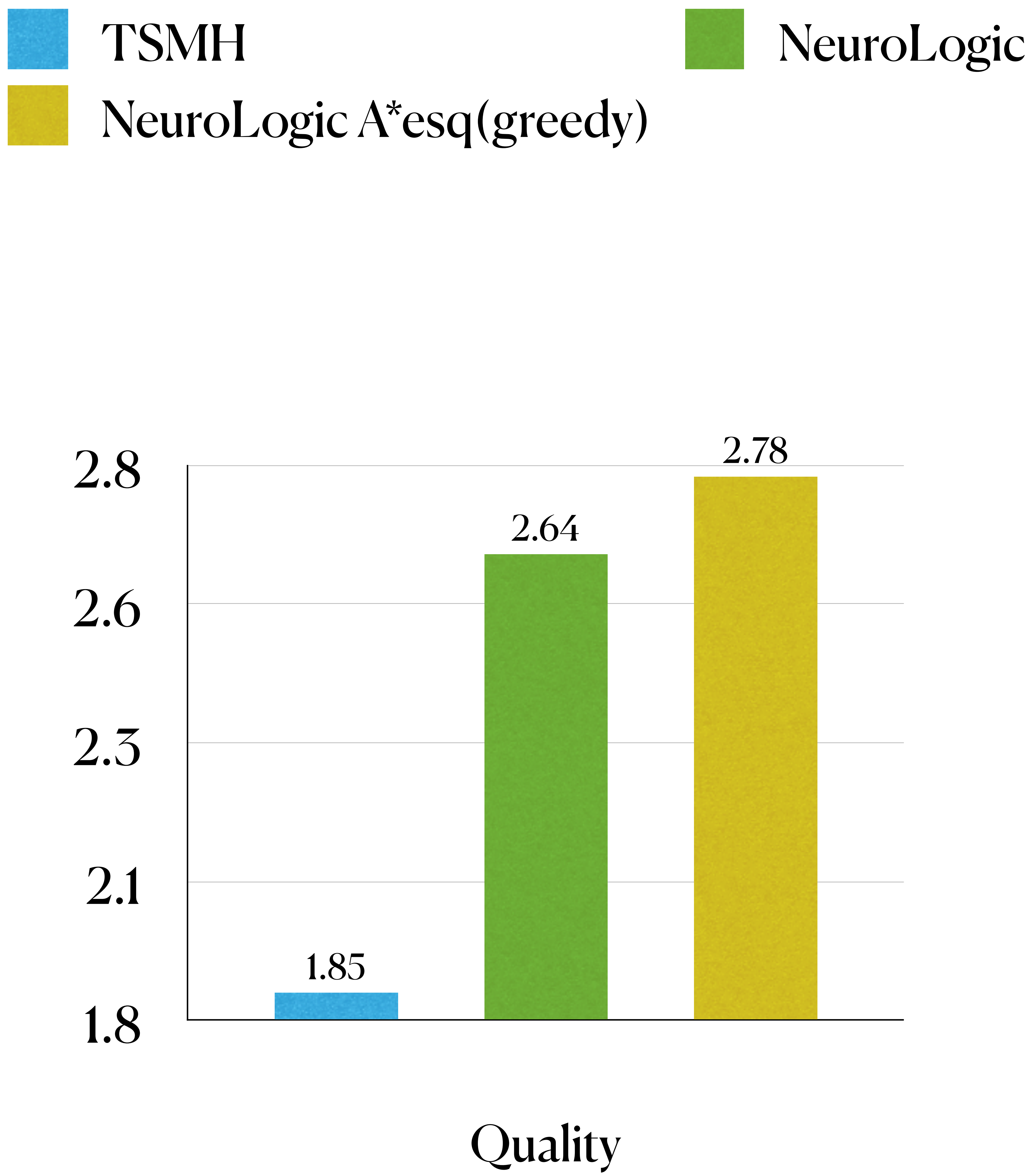
Human evaluation | CommonGen

(Lin et al., 2020)

Fine-tuned GPT-2

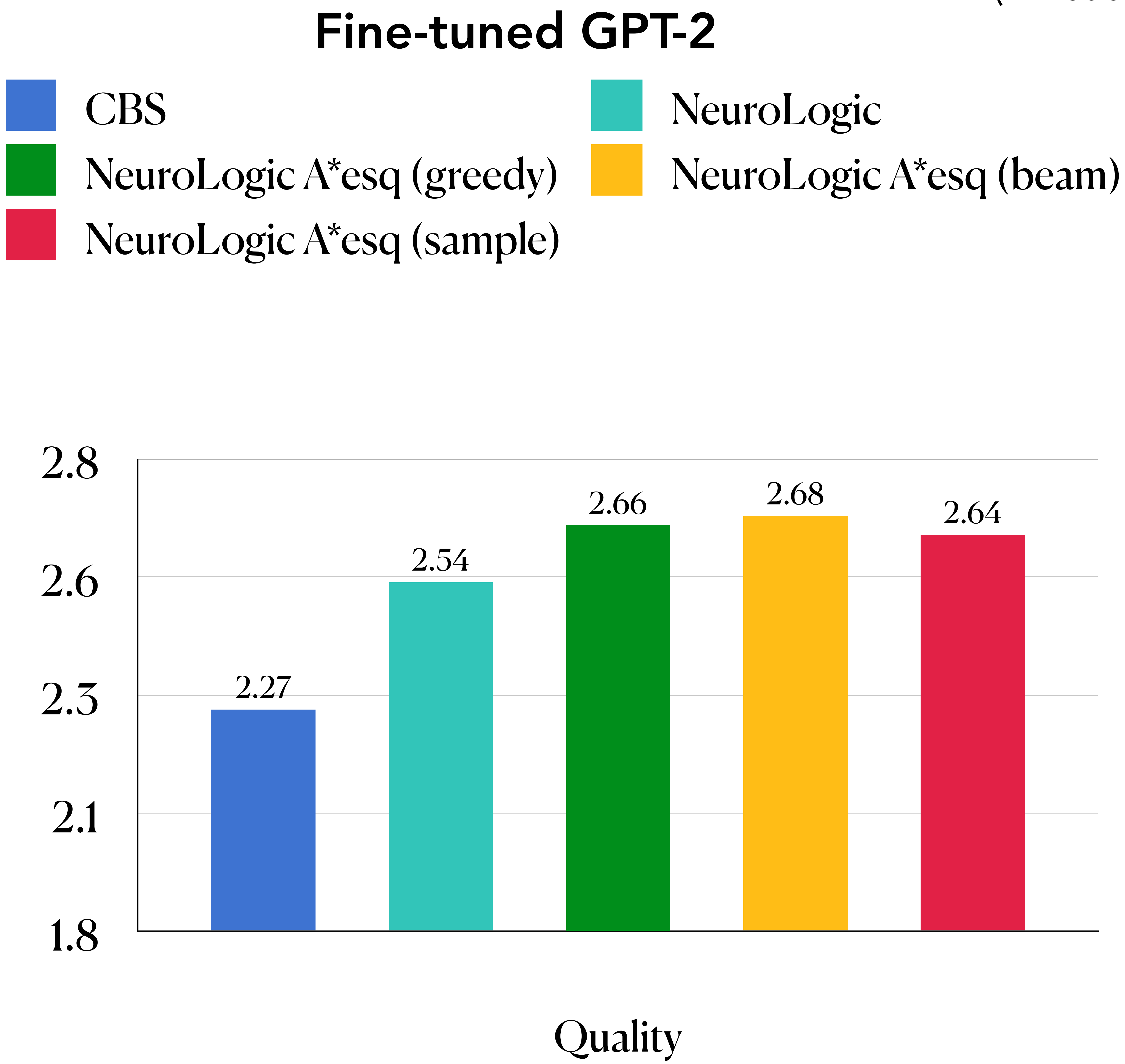


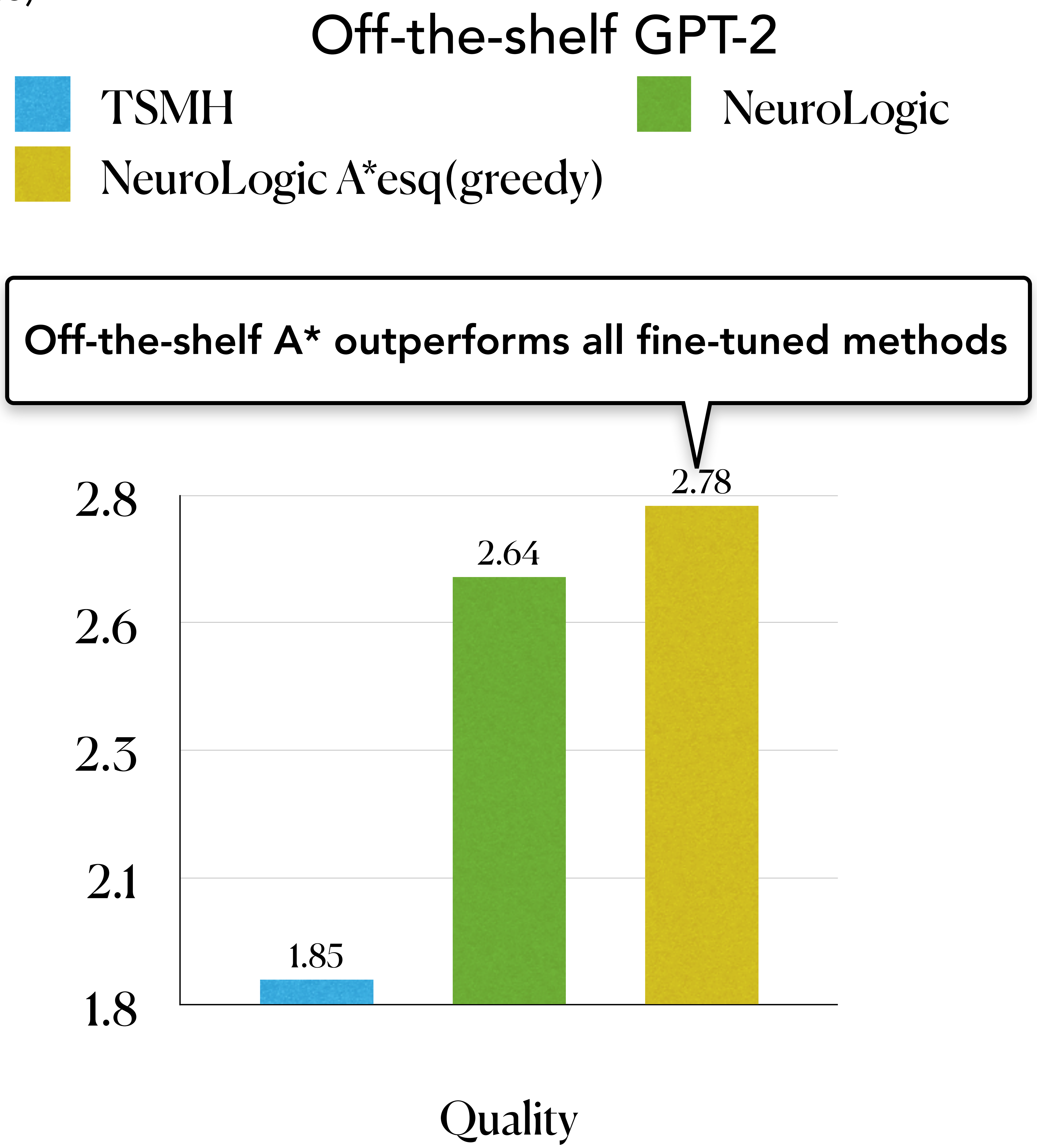
Off-the-shelf GPT-2



Human evaluation | CommonGen

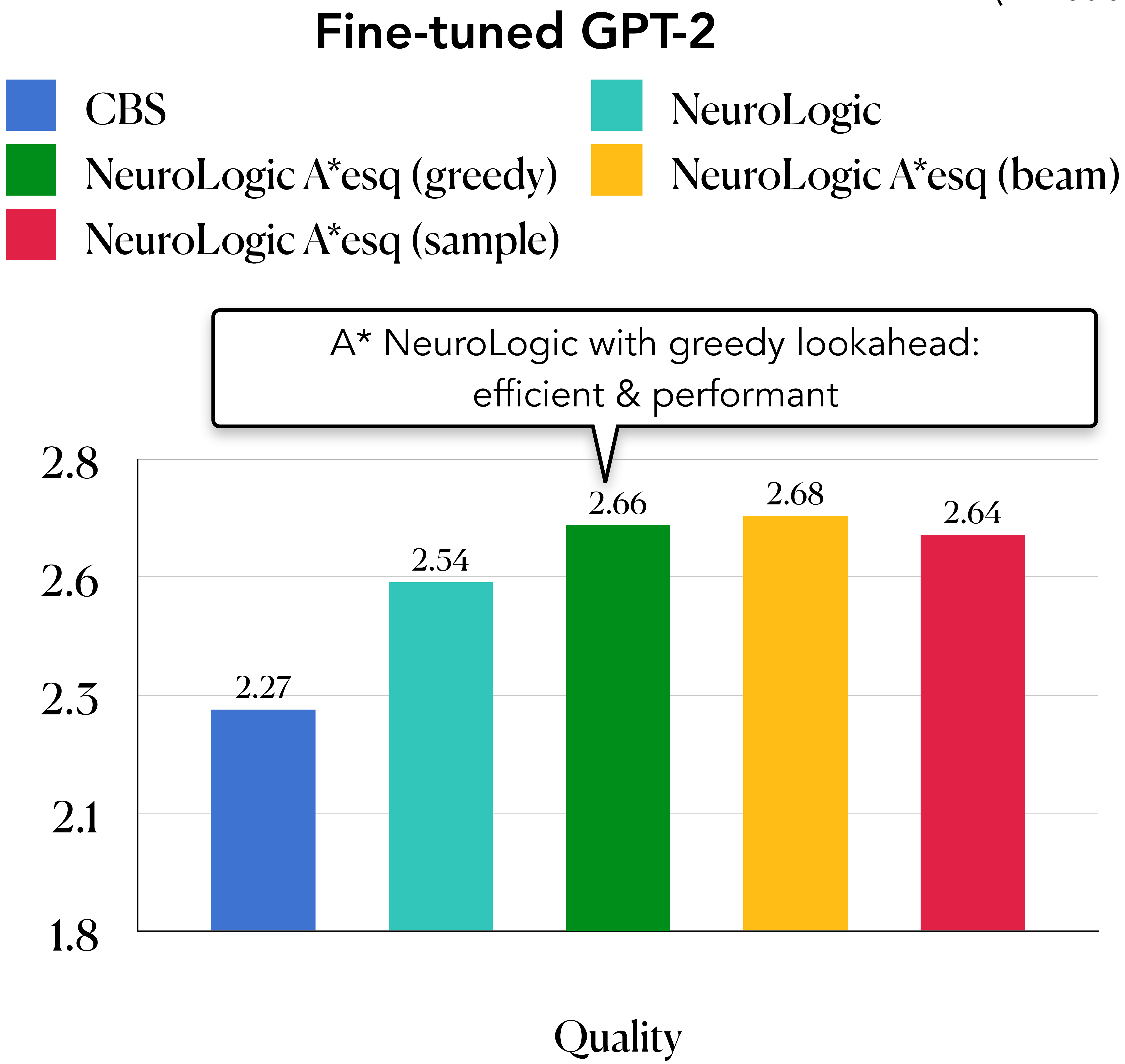
(Lin et al., 2020)

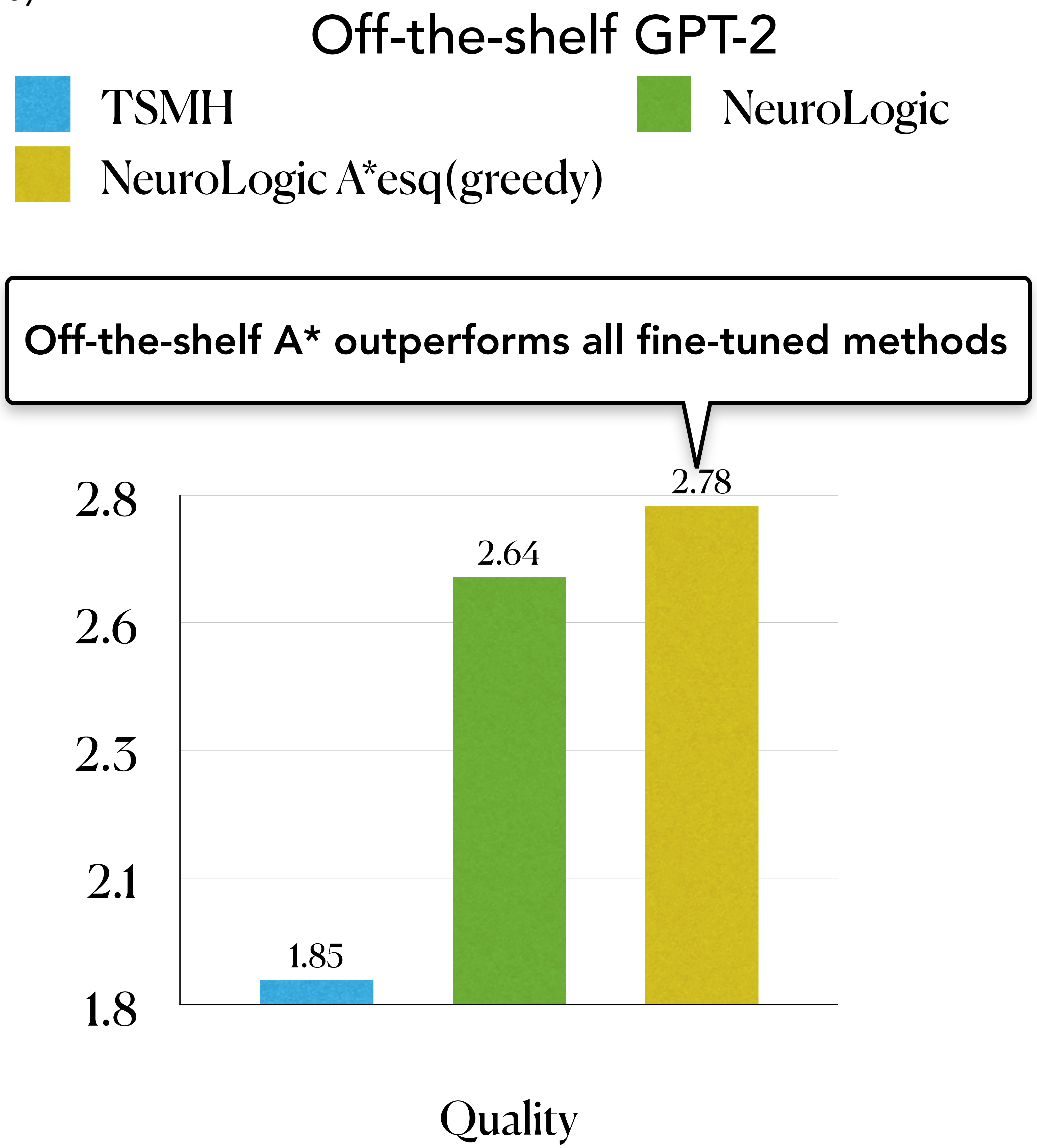




Human evaluation | CommonGen

(Lin et al., 2020)

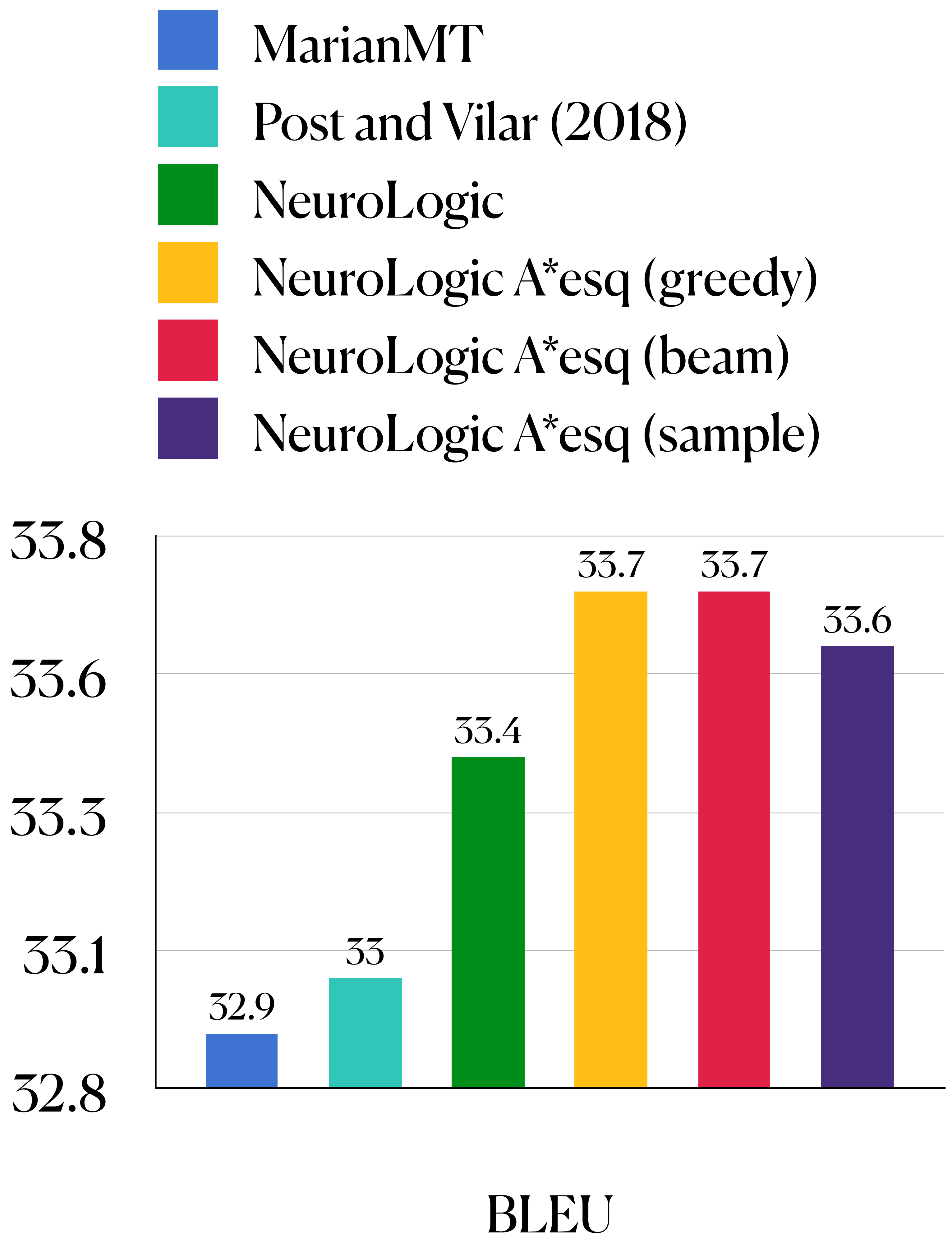




Enables many constrained generation tasks

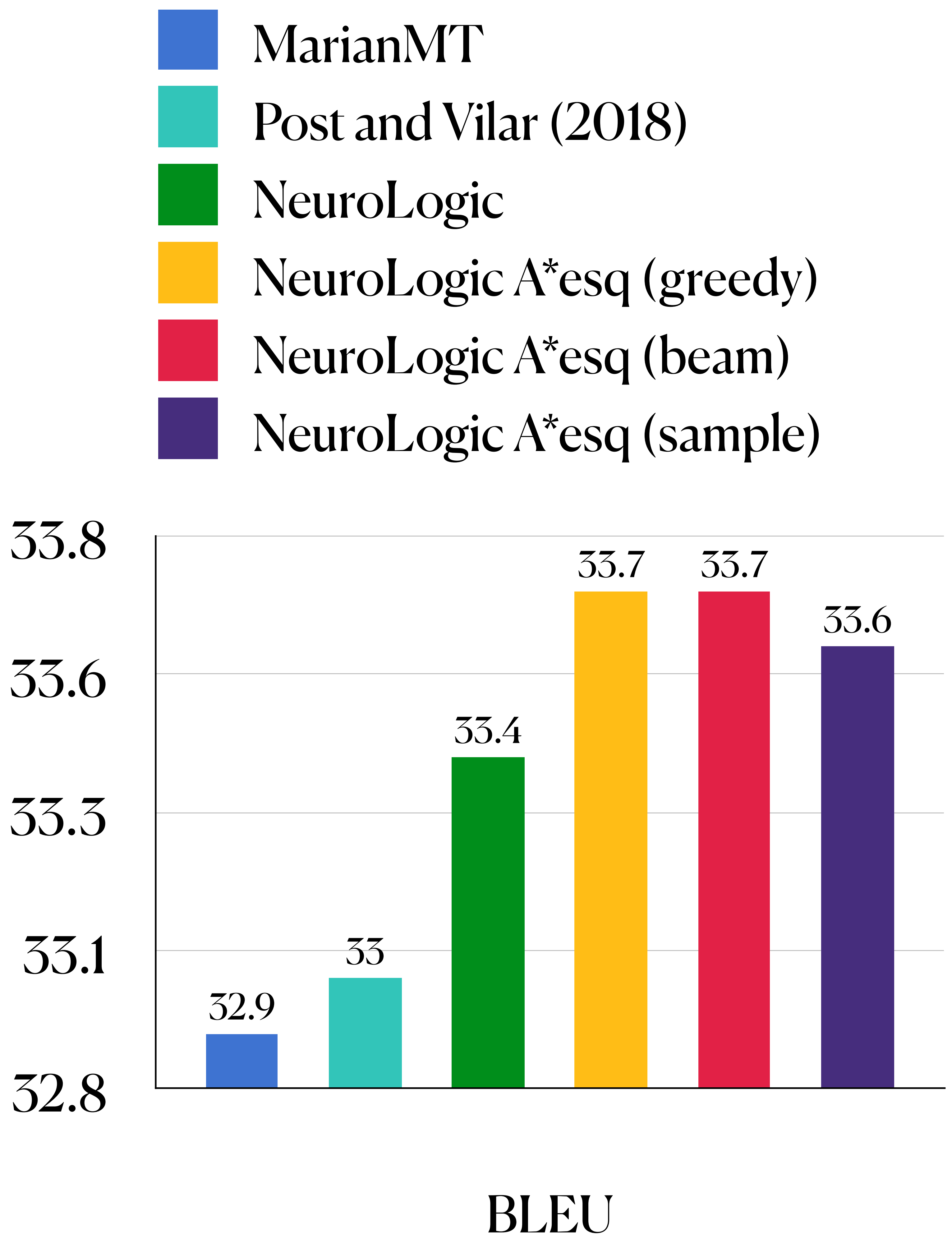
Enables many constrained generation tasks

Constrained MT
(Dinu et al., 2019)

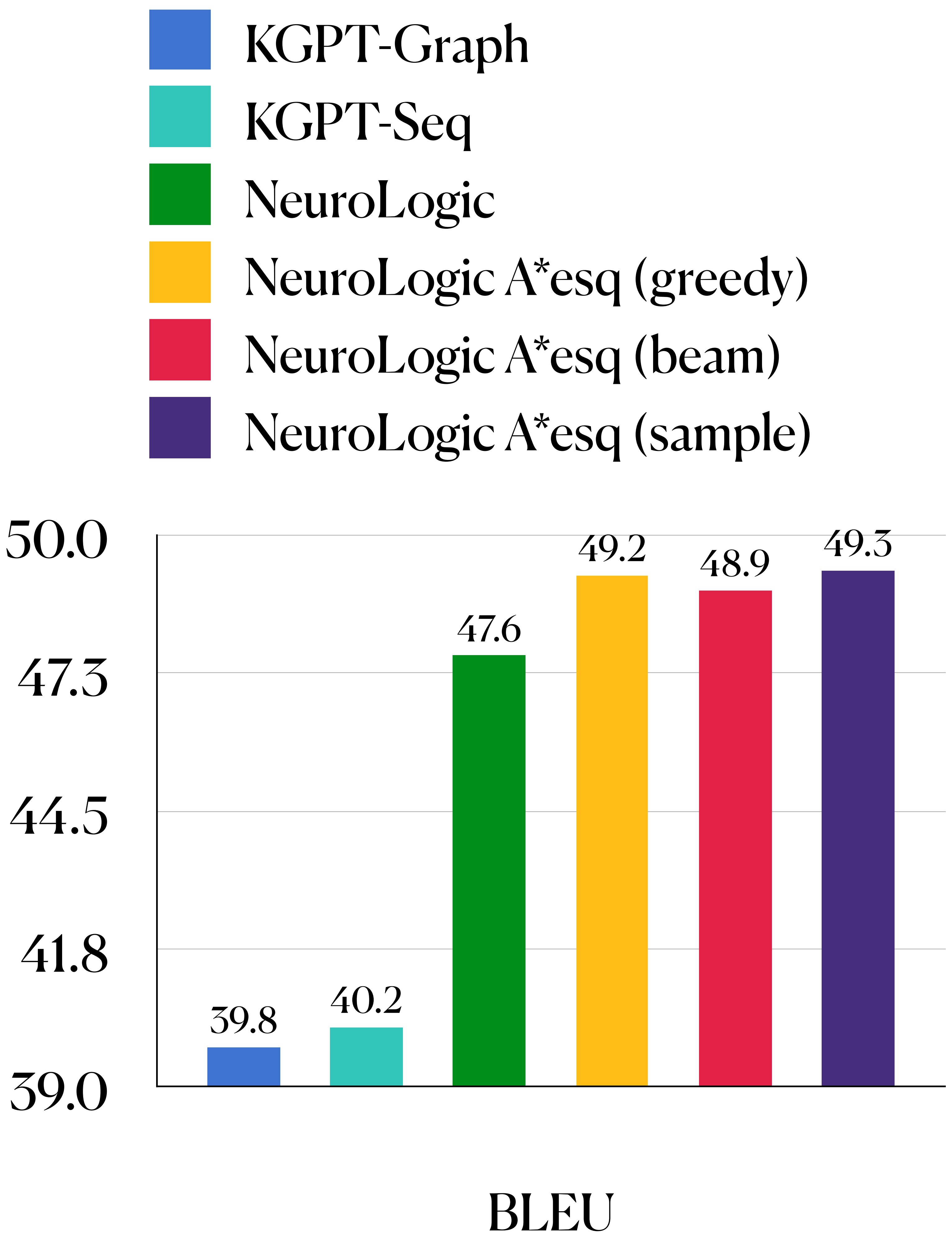


Enables many constrained generation tasks

Constrained MT
(Dinu et al., 2019)

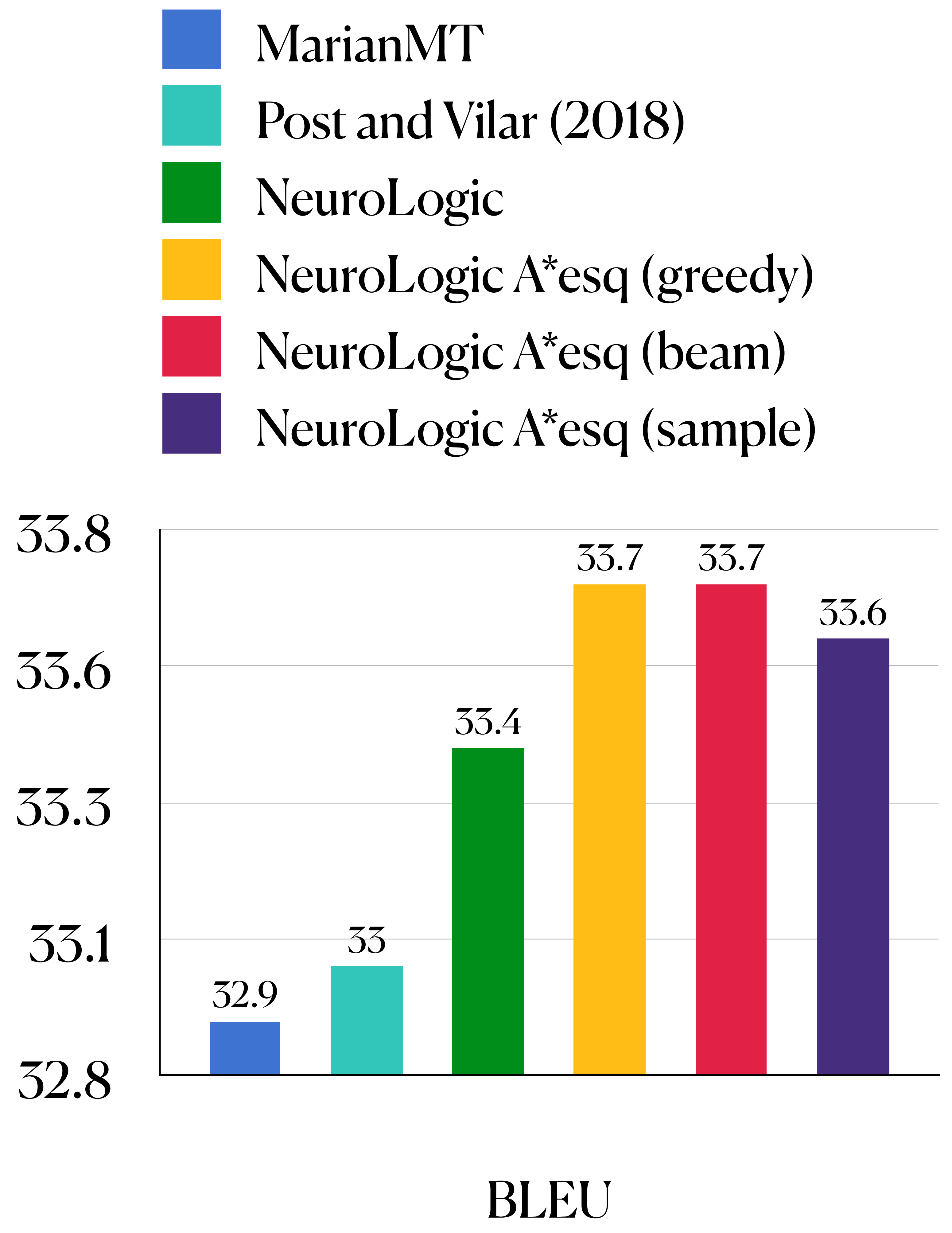


Few-Shot E2ENLG
(Chen et al., 2020)

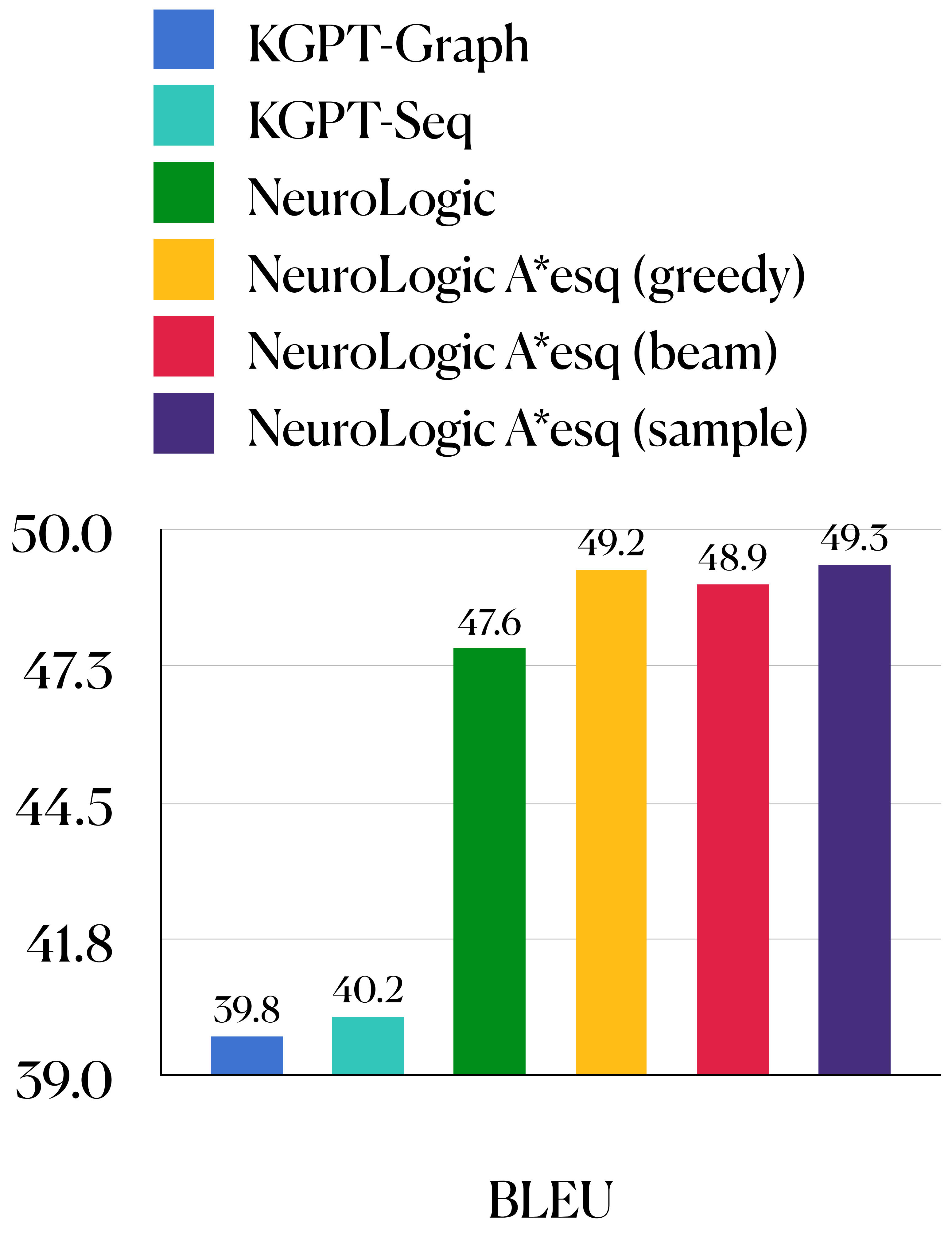


Enables many constrained generation tasks

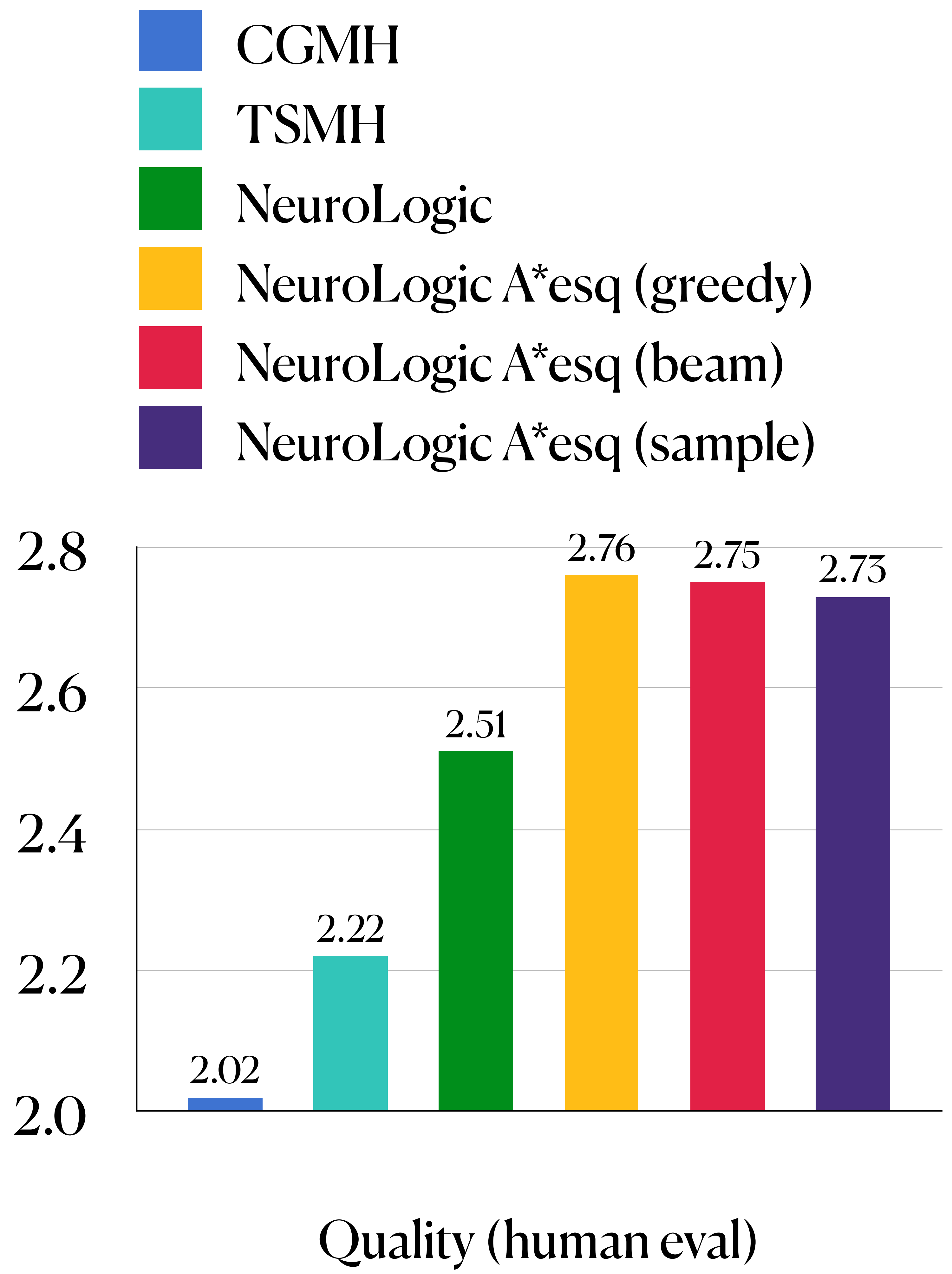
Constrained MT
(Dinu et al., 2019)



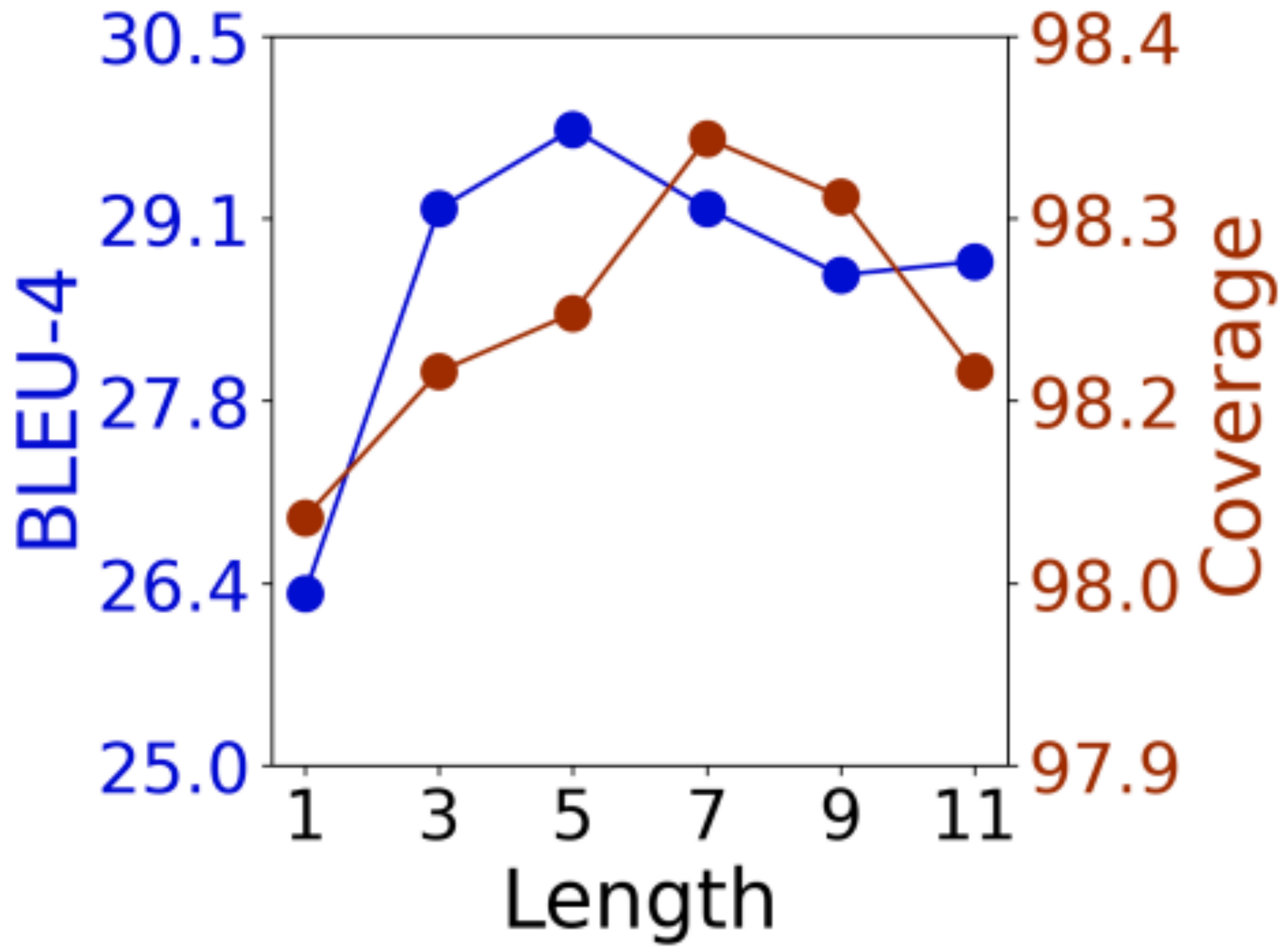
Few-Shot E2ENLG
(Chen et al., 2020)



Question Generation
(Zhang et al., 2020)



- Greedy lookahead length (CommonGen)



- Improves at varying amounts of training data

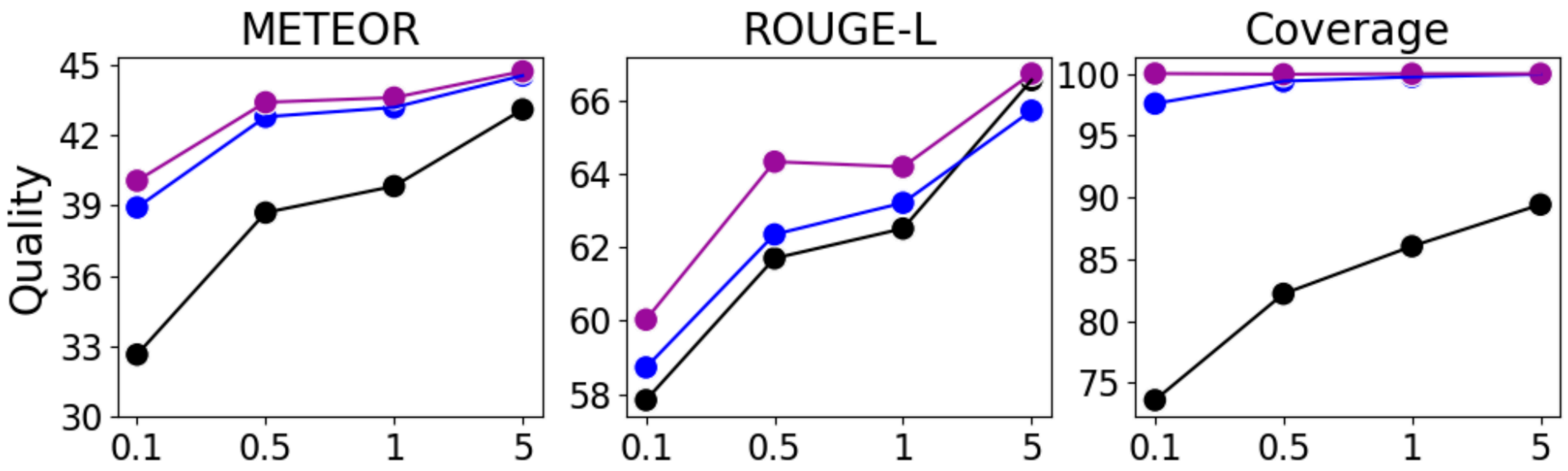
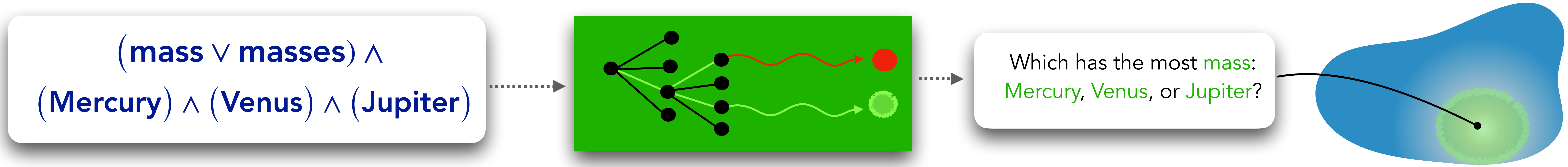


Figure 3: Performance (y-axis) of supervised GPT-2 on E2ENLG, with a varying amount of training data for supervision (x-axis). The **purple**, **blue**, and **black** line denote decoding with NEUROLOGIC★, NEUROLOGIC and conventional beam search respectively.

Constrained generation through *discrete* inference

A* Neurologic

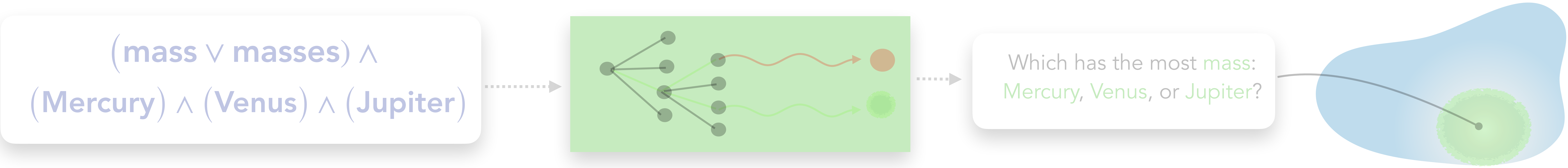
- **Constraints:** expressive class of lexical constraints
- **Search**: discrete with future approximation
- **Enables**: constraints without fine-tuning, better fine-tuned performance



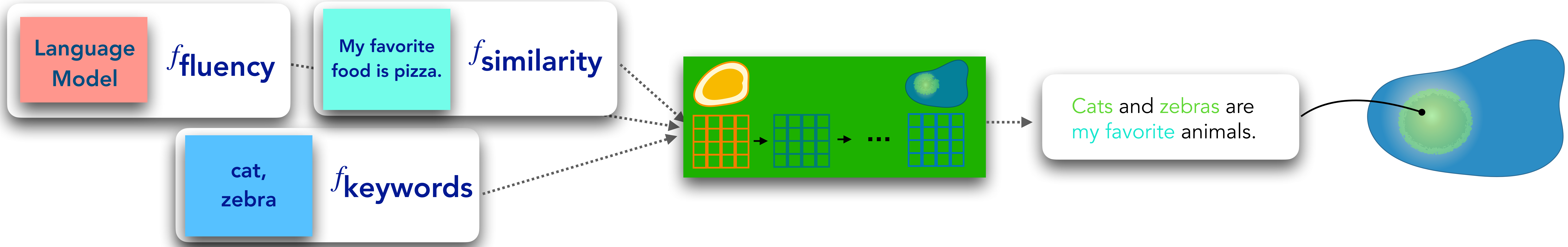
NeuroLogic A*esque Decoding:
Constrained Text Generation with Lookahead Heuristics
[arxiv:2112.08726]
[github.com/GloriaXimingLu/star_neurologic]

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

UNIVERSITY of WASHINGTON


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.

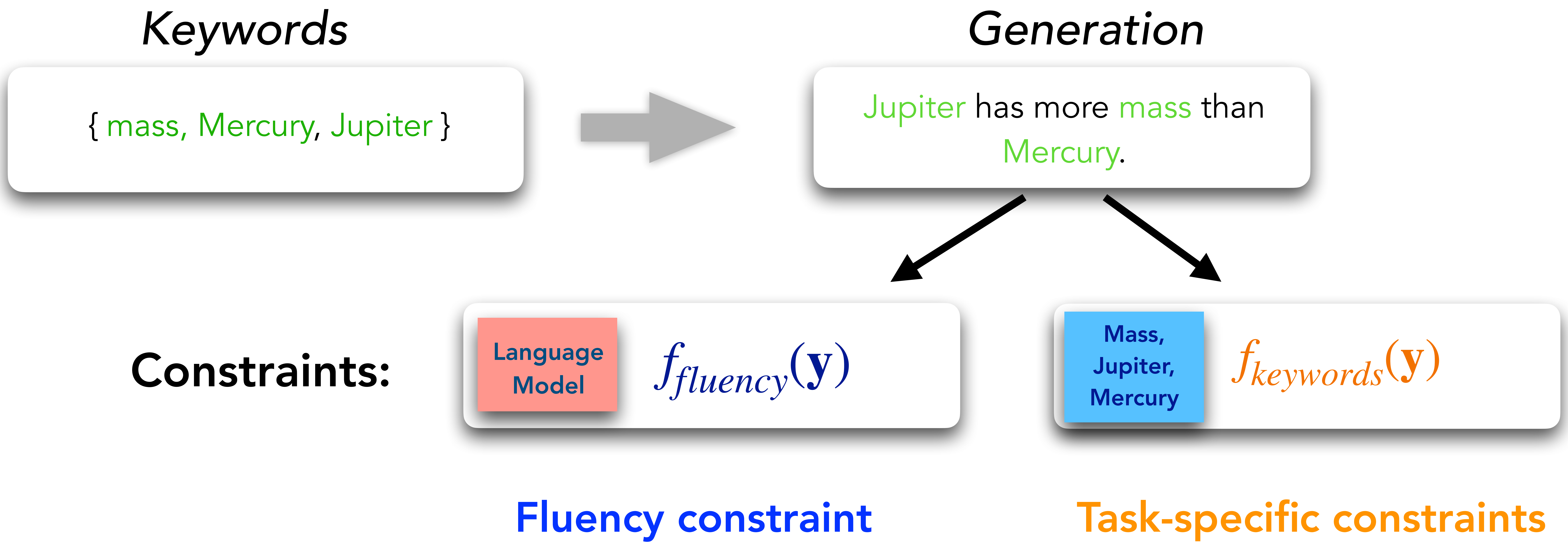
Constraints:

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.

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:

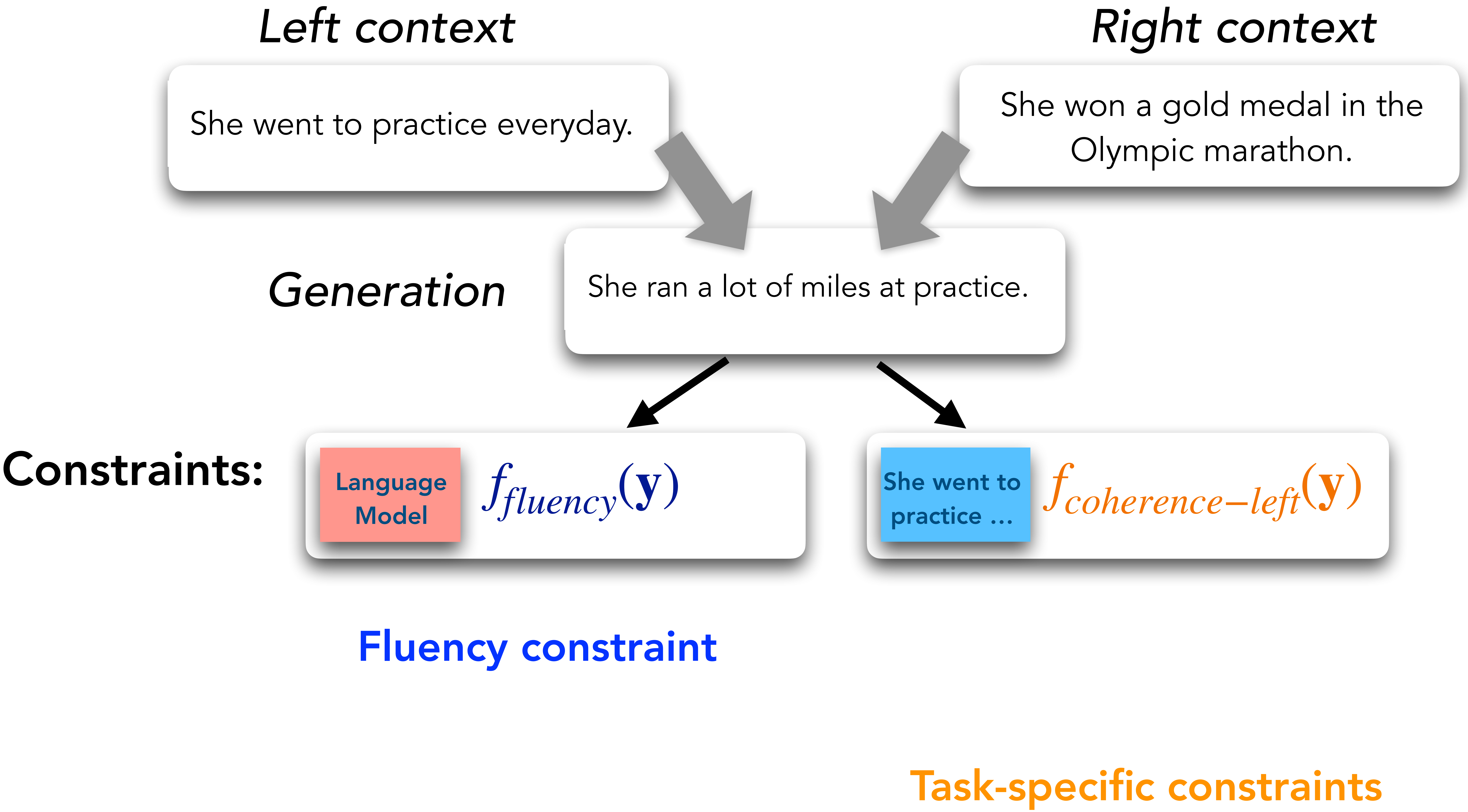
Language Model $f_{fluency}(\mathbf{y})$

Fluency constraint

Text infilling / abductive reasoning

AbductiveNLG

(Bhagavatula et al., 2020)



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

She went to practice ... $f_{coherence-left}(\mathbf{y})$

She won a gold ... $f_{coherence-right}(\mathbf{y})$

Fluency constraint

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)

~~The law student~~

joined a prestigious law firm after graduating.

The **medical** student

Keep Similar

Text similarity / counterfactual reasoning

TimeTravel

(Qin et al., 2019)

~~The law student~~ joined a prestigious law firm after graduating.

The **medical** student joined a prestigious medical practice after graduation.

Keep Similar

Generation

Text similarity / counterfactual reasoning

TimeTravel

(Qin et al., 2019)

~~The law student~~	joined a prestigious law firm after graduating.
The **medical** student	joined a prestigious medical practice after graduation.

Keep Similar

Generation

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

Fluency constraint

Text similarity / counterfactual reasoning

TimeTravel
(Qin et al., 2019)

~~The law student~~	joined a prestigious law firm after graduating.
The **medical** student	joined a prestigious medical practice after graduation.

Keep Similar

Generation

Constraints:

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

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

Fluency constraint

Task-specific constraints

Text similarity / counterfactual reasoning

TimeTravel

(Qin et al., 2019)

~~The law student~~ joined a prestigious law firm after graduating.

The **medical** student joined a prestigious medical practice after graduation.

Keep Similar

Generation

Constraints:

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

Fluency constraint

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

Joined a prestigious ... $f_{similarity}(\mathbf{y}, \mathbf{y}_{*})$

Task-specific constraints

Constrained generation as sampling from an energy-based model

Fluency constraint

Task-specific constraints

Energy function: $E(\mathbf{y}) = f_{fluency}(\mathbf{y}) + f_1(\mathbf{y}) + f_2(\mathbf{y}) + \ldots$

Constrained generation as sampling from an energy-based model

Fluency constraint

Task-specific constraints

Energy function: $$E(\mathbf{y}) = f_{fluency}(\mathbf{y}) + f_1(\mathbf{y}) + f_2(\mathbf{y}) + \ldots$$

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

Constrained generation as sampling from an energy-based model

Fluency constraint

Task-specific constraints

Energy function: $E(\mathbf{y}) = f_{fluency}(\mathbf{y}) + f_1(\mathbf{y}) + f_2(\mathbf{y}) + \ldots$

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

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

Sampling from an energy-based model

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

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

Sampling from an energy-based model

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

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

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

Sampling from an energy-based model

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

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

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

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

Sampling from an energy-based model

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

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

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

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

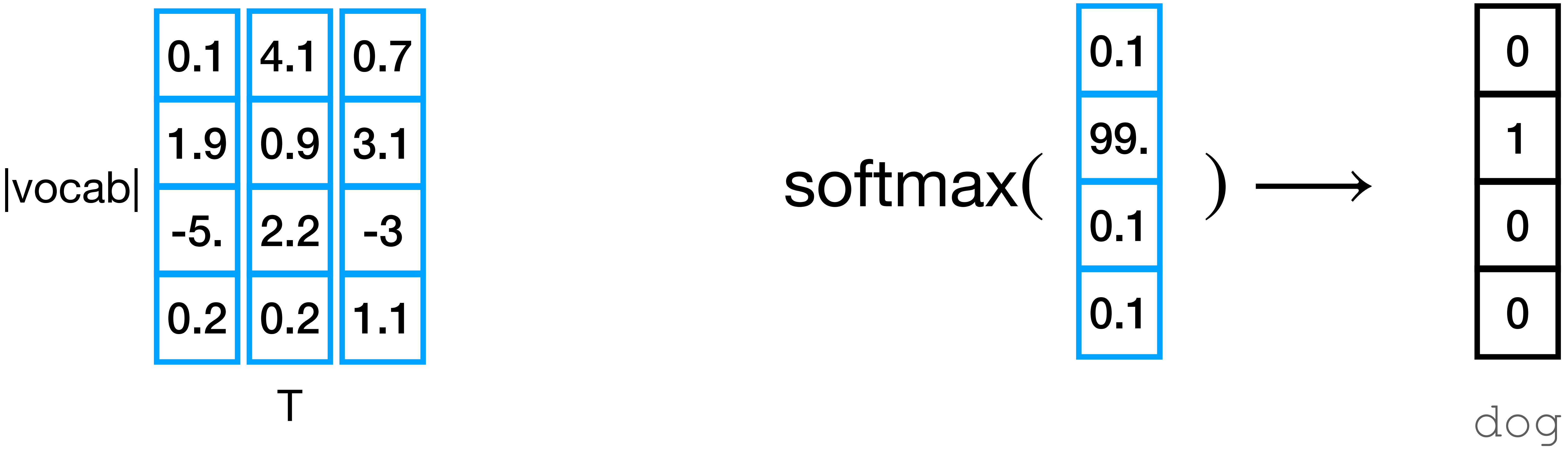
- Define energy over "**soft sequence**" of continuous vectors:
  - $\tilde{\mathbf{y}} = (\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_T)$, where $\tilde{\mathbf{y}}_t \in \mathbf{R}^{vocab}$

$$\begin{array}{cc} |\text{vocab}| & \begin{bmatrix} 0.1 & 4.1 & 0.7 \\ 1.9 & 0.9 & 3.1 \\ -5. & 2.2 & -3 \\ 0.2 & 0.2 & 1.1 \end{bmatrix} \\ & \text{T} \end{array}$$

Sampling from an energy-based model

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

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



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

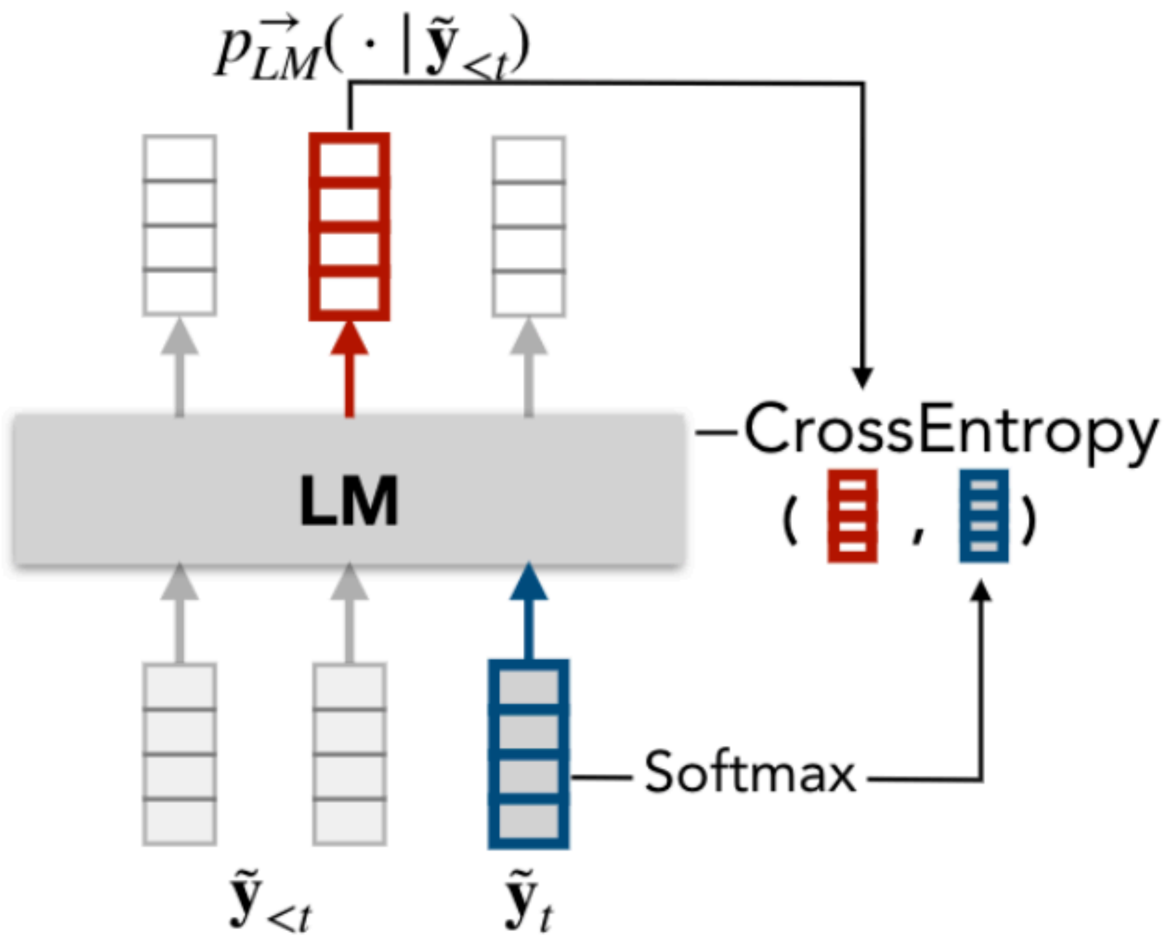
Sampling from an energy-based model

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

- Constraints as **differentiable functions**

Language Model $f_{fluency}(\tilde{\mathbf{y}})$

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



Sampling from an energy-based model

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

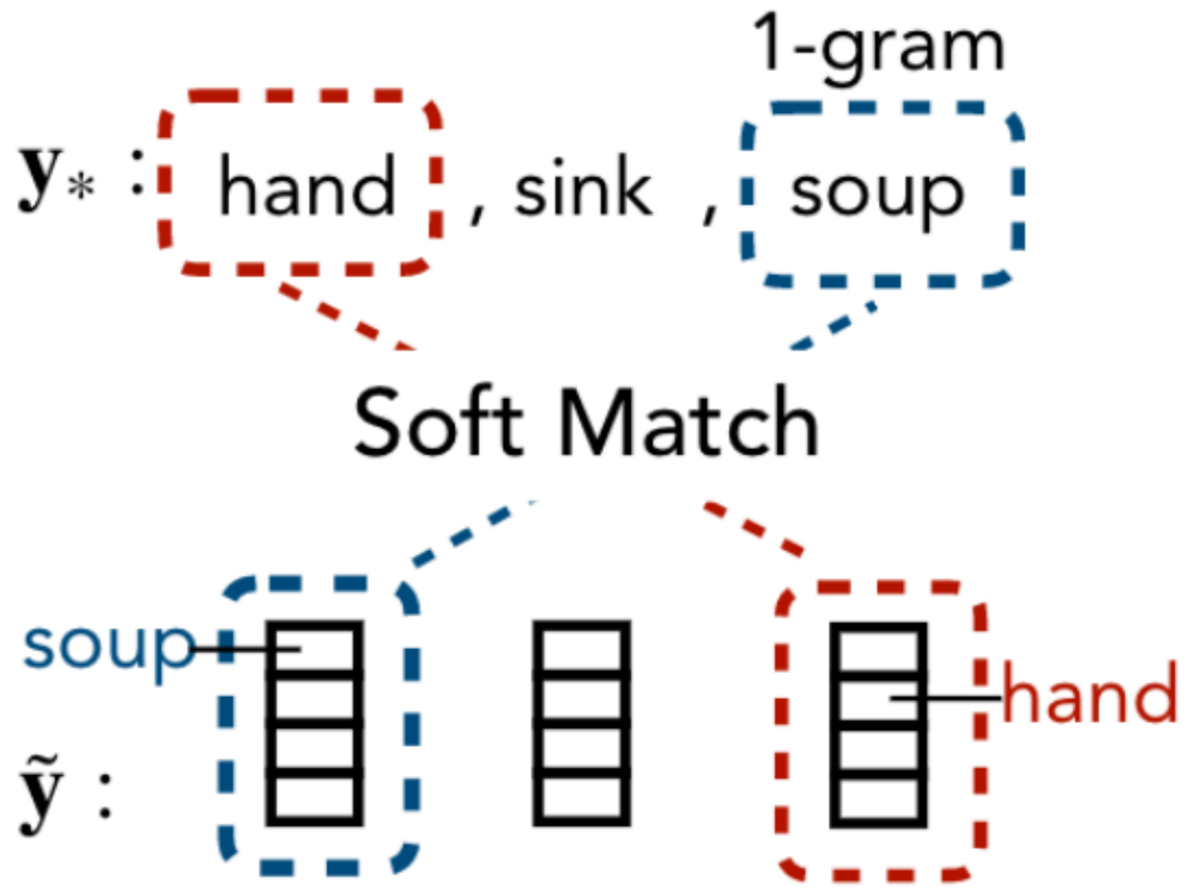
- Constraints as **differentiable functions**

Mass, Jupiter, Mercury

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

Joined a prestigious …

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

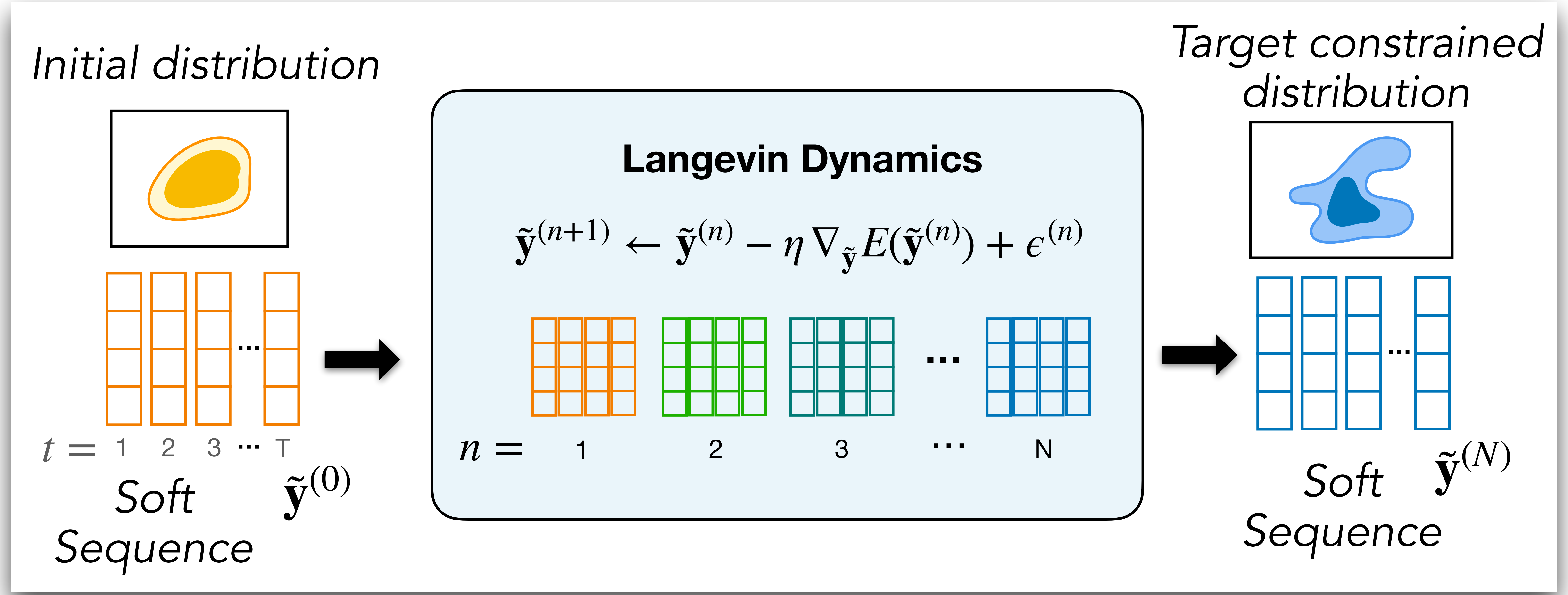


$$f_{\text{sim}}(\tilde{\mathbf{y}}; \mathbf{y}_*) = \texttt{ngram-match}(\tilde{\mathbf{y}}, \mathbf{y}_*),$$

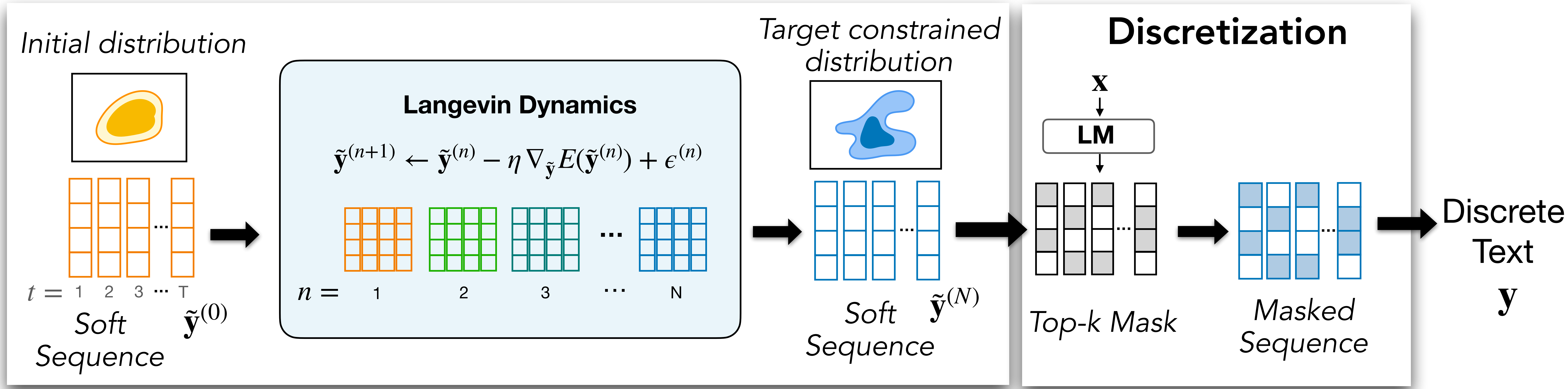
(Liu et al., 2021)

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

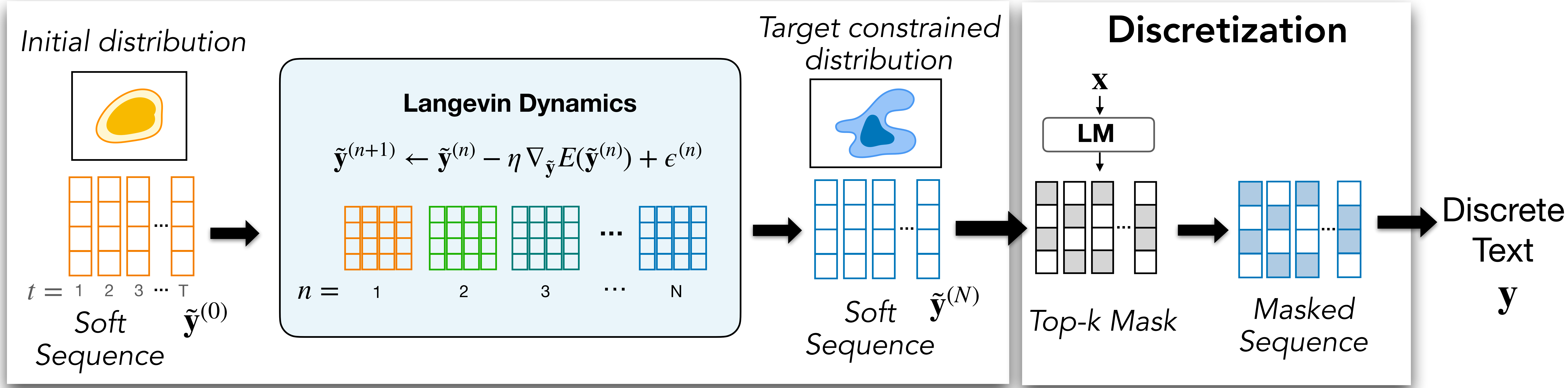
Specify energy $E(\tilde{\mathbf{y}}) = \sum_i f_i(\tilde{\mathbf{y}})$, then:



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:

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

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:

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

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**

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

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**

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

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)

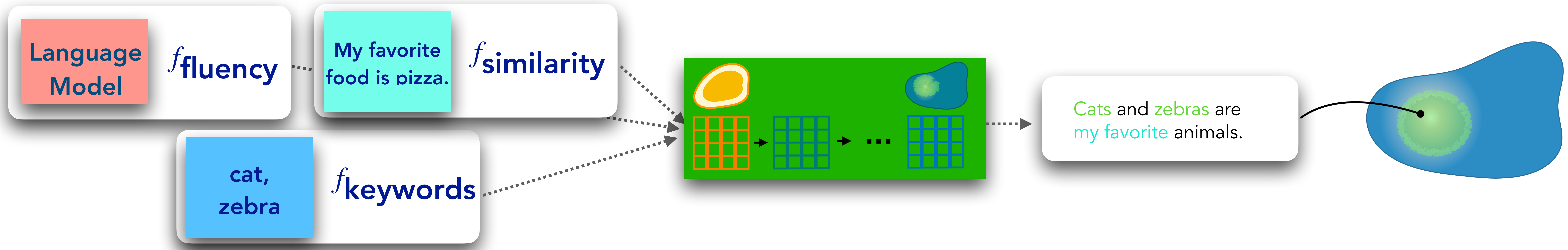
Models	Grammar	Left-coher. (x-y)	Right-coher. (y-z)	Overall-coher. (x-y-z)
COLD (Full)	4.17	3.96	**2.88**	**2.83**
COLD $-f_{\text{sim}}$	4.54	3.82	2.73	2.69
COLD $-f_{\text{LM}}^{\leftarrow}$	4.35	3.97	2.84	2.80
COLD $-f_{\text{pred}}$	**4.61**	**4.07**	2.75	2.77

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
github.com/qkaren/COLD_decoding

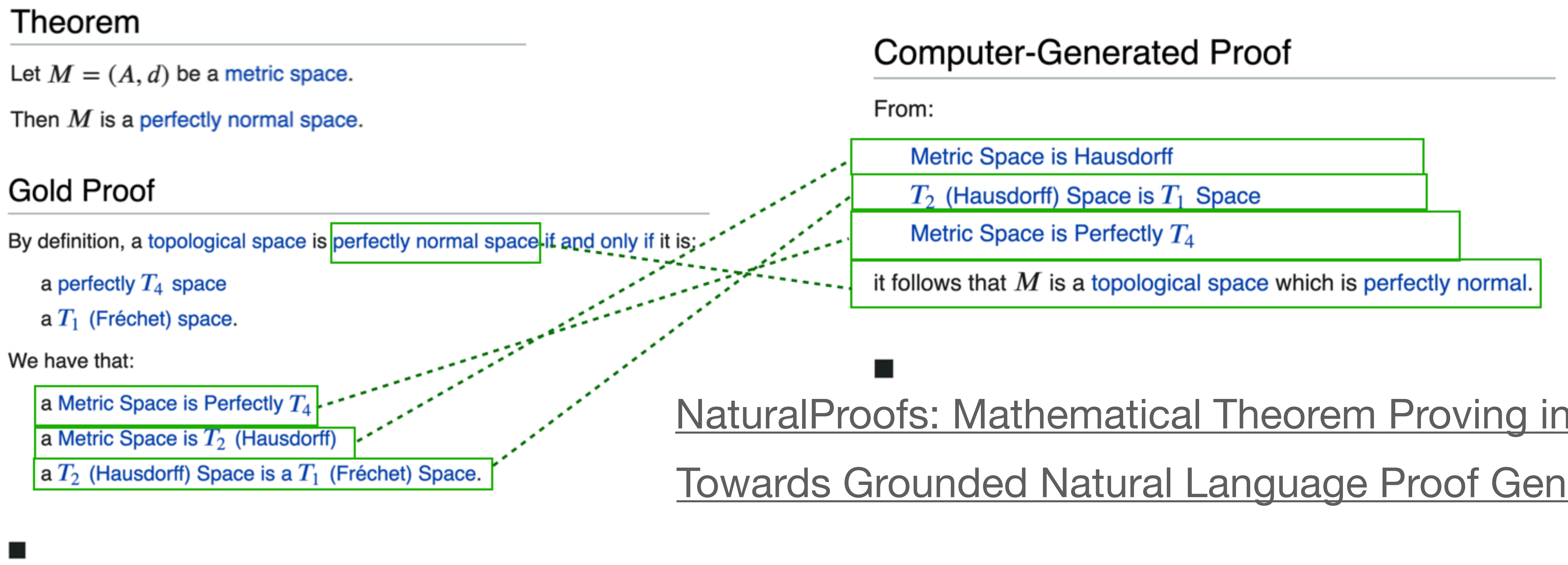
Constrained generation

Looking ahead

Constrained generation

Looking ahead

- **Grounded generation**



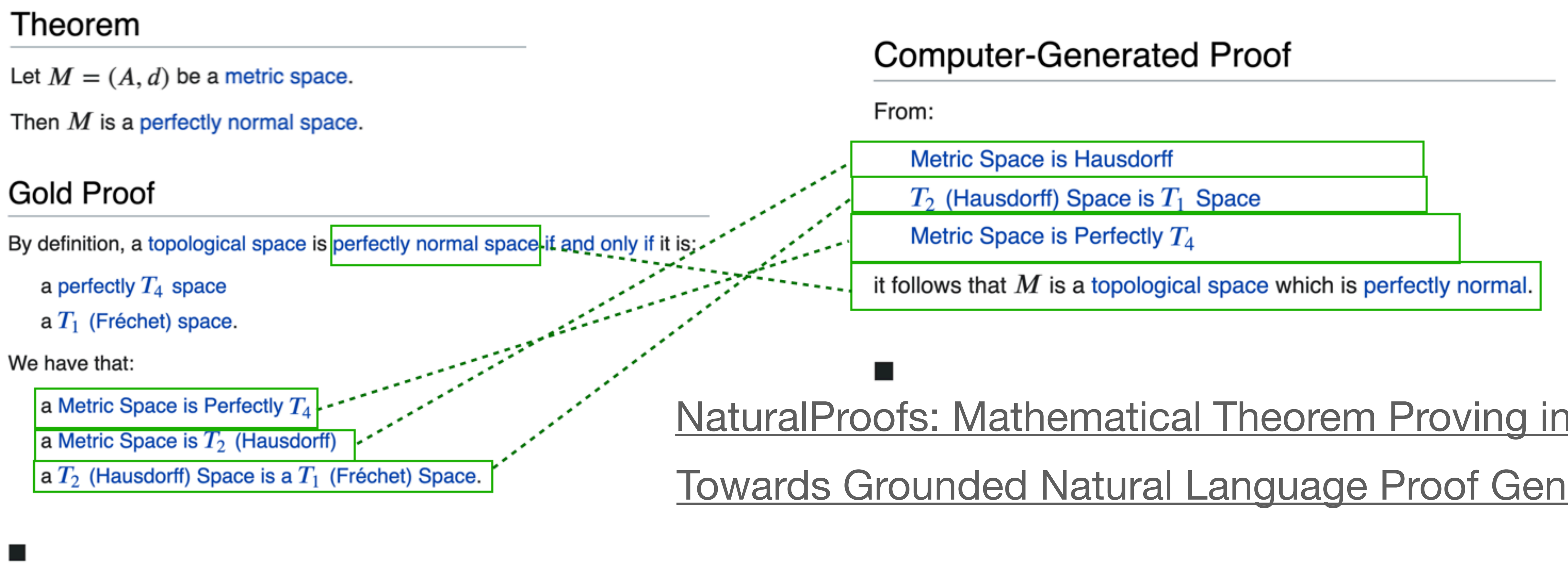
[NaturalProofs: Mathematical Theorem Proving in Natural Language](#)

[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 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

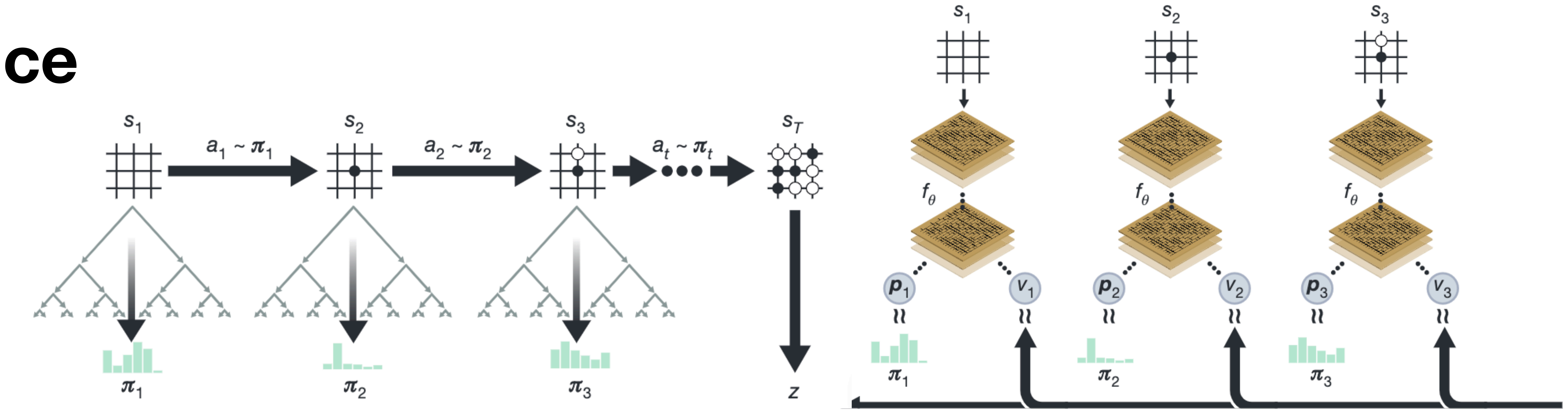
it follows that M is a topological space which is perfectly normal.

■

NaturalProofs: Mathematical Theorem Proving in Natural Language

Towards Grounded Natural Language Proof Generation (Work in Progress)

- **Joint learning & inference**



[Silver et al 2017]

Thanks for your attention!