Sequence Modeling: Generation

EECS 183/283a: Natural Language Processing

Generation

- Given a language model $p(\overline{X}) \in \Delta^{\mathcal{V}^+}$, how do we get a plausible sequence out of it?

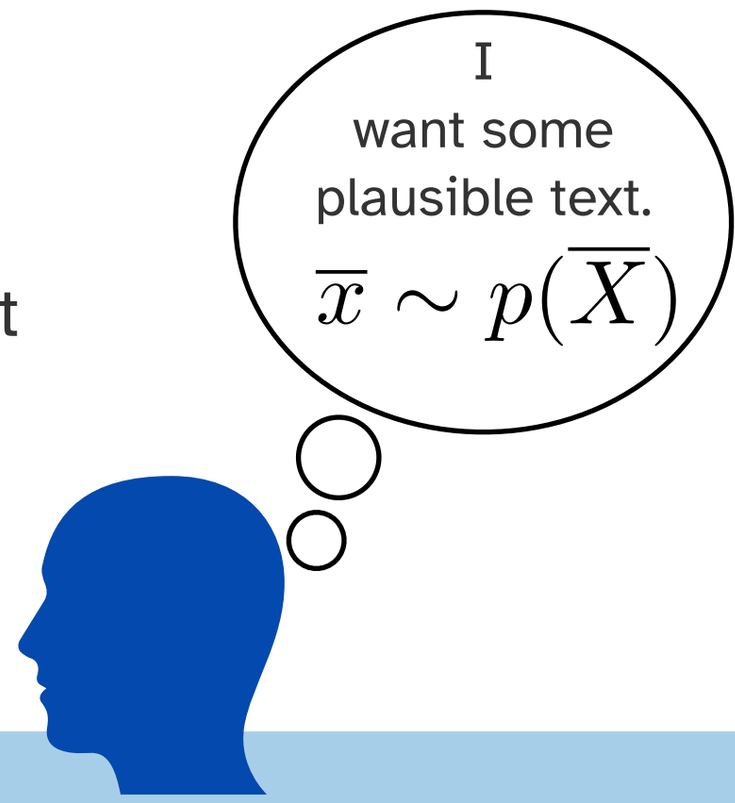
- Autoregressive language model:

$$p(\overline{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

- We can sample directly from this approximation

- We can adjust this distribution, then sample from it

- We can try to find the *most* plausible sequence in $\mathcal{V}^+$



Sequential Generation

$$p(\overline{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

- As we generate, we build our output sequence $\overline{x}$, which starts as an empty sequence $\overline{x} = \langle\ \rangle$

Sequential Generation

$$p(\overline{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

- As we generate, we build our output sequence $\overline{x}$, which starts as an empty sequence $\overline{x} = \langle\ \rangle$
- At each step i, we choose an item from the vocabulary $\mathcal{V}$ by performing some operation on the local probability distribution $p(X_i \mid x_1, \ldots, x_{i-1}) \in \Delta^{\mathcal{V}}$

Sequential Generation

$$p(\overline{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

- As we generate, we build our output sequence $\overline{x}$, which starts as an empty sequence $\overline{x} = \langle\,\rangle$
- At each step i, we choose an item from the vocabulary $\mathcal{V}$ by performing some operation on the local probability distribution $p(X_i \mid x_1, \ldots, x_{i-1}) \in \Delta^{\mathcal{V}}$
- Then, we append this to our running sequence $\overline{x} \leftarrow \overline{x} + \langle x_i \rangle$

Sequential Generation

$$p(\overline{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

- As we generate, we build our output sequence $\overline{x}$, which starts as an empty sequence $\overline{x} = \langle\,\rangle$
- At each step i, we choose an item from the vocabulary $\mathcal{V}$ by performing some operation on the local probability distribution $p(X_i \mid x_1, \ldots, x_{i-1}) \in \Delta^{\mathcal{V}}$
- Then, we append this to our running sequence $\overline{x} \leftarrow \overline{x} + \langle x_i \rangle$
- If we ever choose `EOS`, we stop generation

Sequential Generation

$$p(\overline{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

- As we generate, we build our output sequence $\overline{x}$, which starts as an empty sequence $\overline{x} = \langle\, \rangle$
- At each step i, we choose an item from the vocabulary $\mathcal{V}$ by performing some **operation** on the local probability distribution $p(X_i \mid x_1, \ldots, x_{i-1}) \in \Delta^{\mathcal{V}}$
- Then, we append this to our running sequence $\overline{x} \leftarrow \overline{x} + \langle x_i \rangle$
- If we ever choose `EOS`, we stop generation
- **Main point:** generation methods differ wrt the **operation** they perform on $p(X_i \mid x_1, \ldots, x_{i-1})$

Recap: Ancestral Sampling

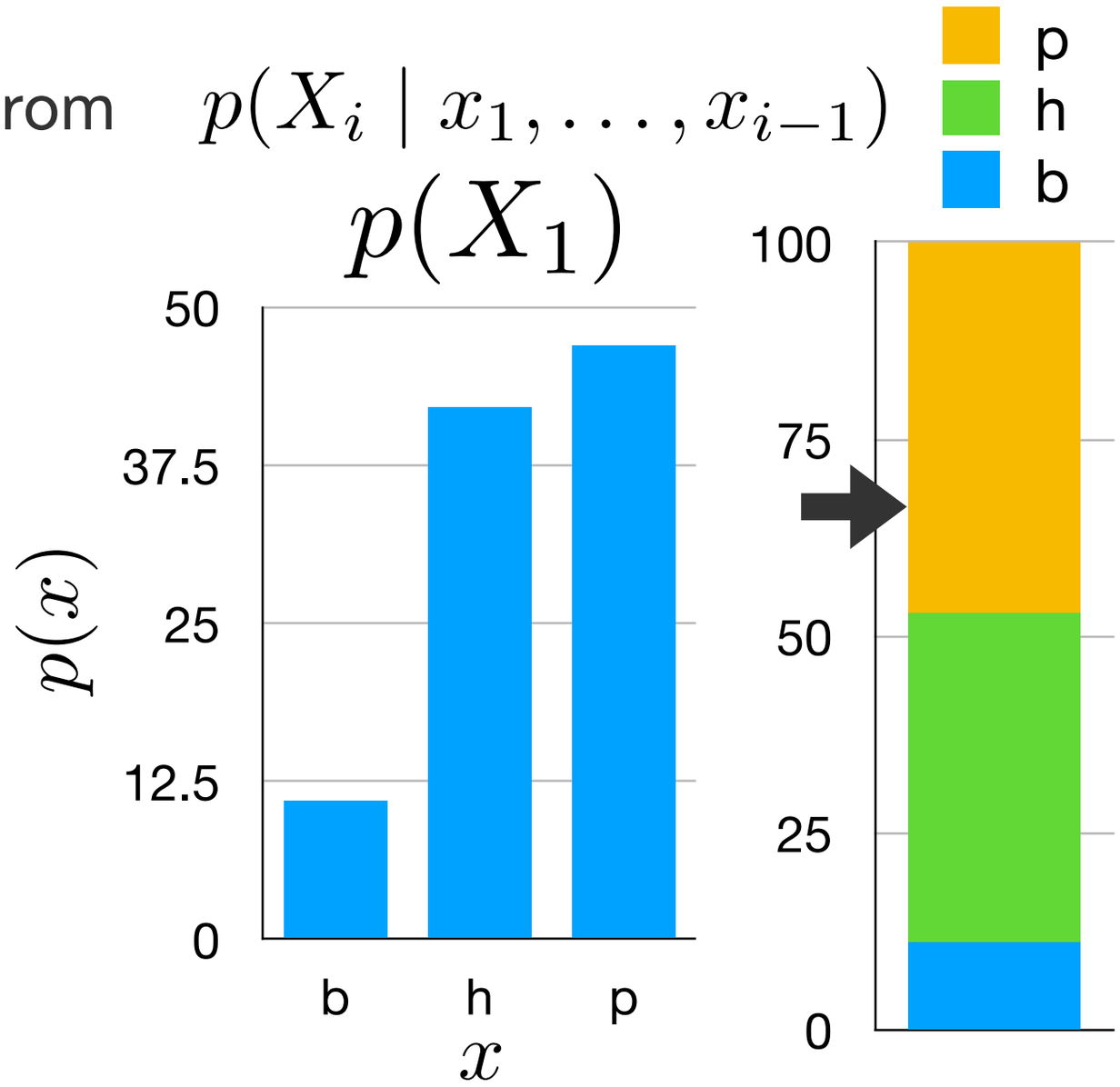
$$p(\overline{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

Operation: sample directly from $p(X_i \mid x_1, \ldots, x_{i-1})$

$\overline{x} = \langle p$

```
number = random(0, 100)
```

number is 65



Recap: Ancestral Sampling

$$p(\overline{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

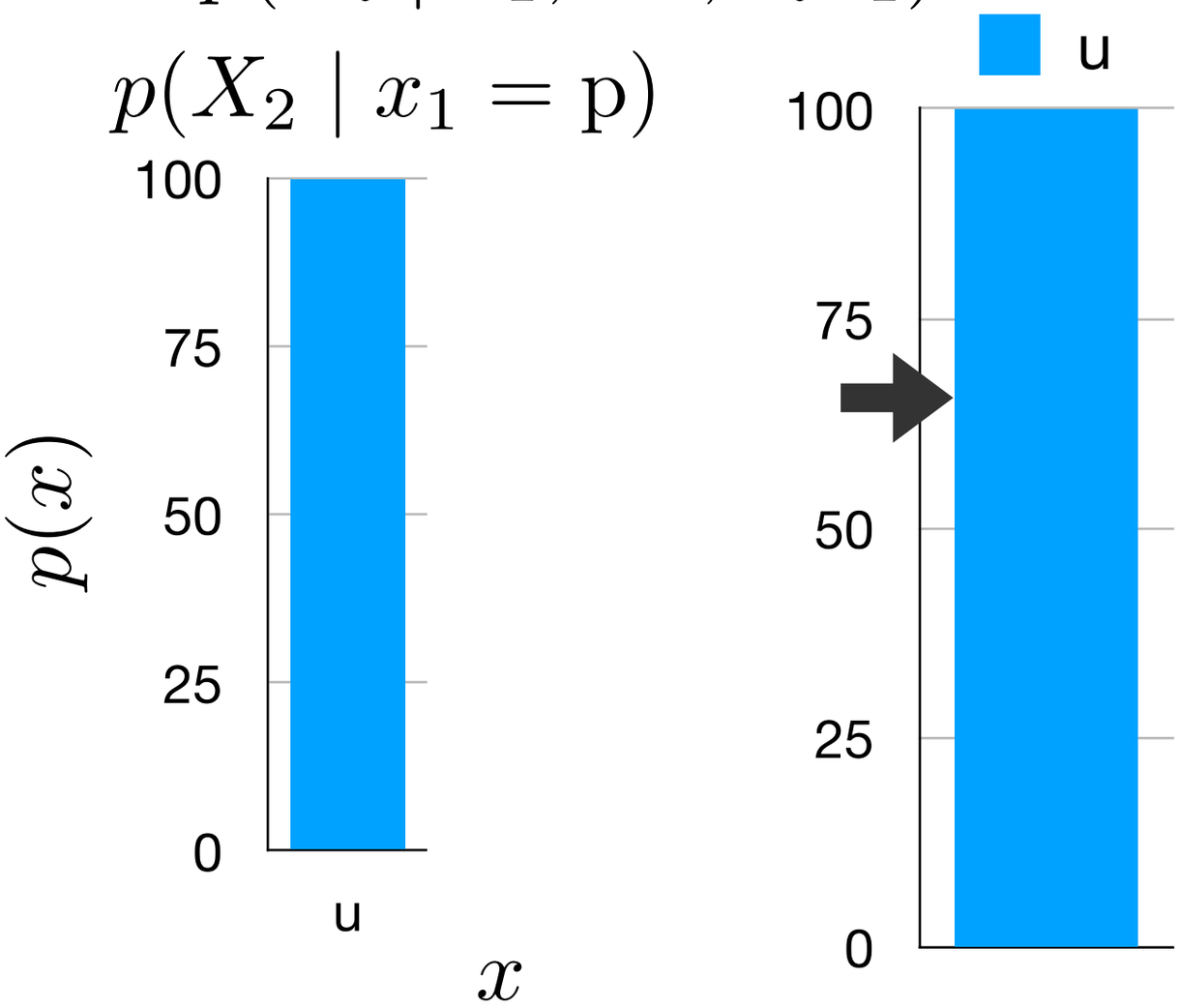
Operation: sample directly from $p(X_i \mid x_1, \ldots, x_{i-1})$

$\overline{x} = \langle p\, u$

```
number = random(0, 100)
```

number is 63

$p(X_2 \mid x_1 = \mathrm{p})$

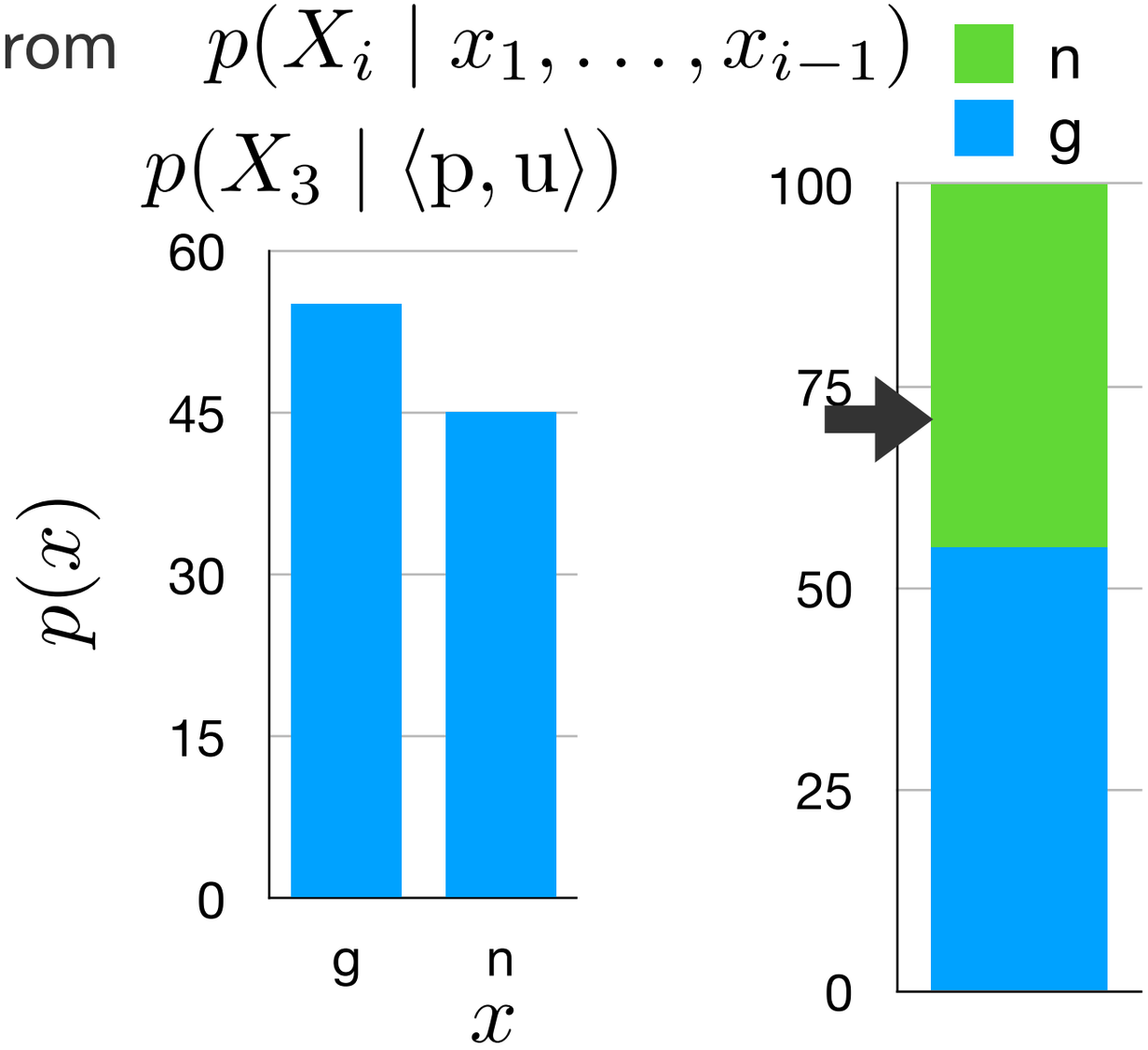


Recap: Ancestral Sampling

$$p(\overline{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

Operation: sample directly from $p(X_i \mid x_1, \ldots, x_{i-1})$

$\overline{x} = \langle p\ u\ n$



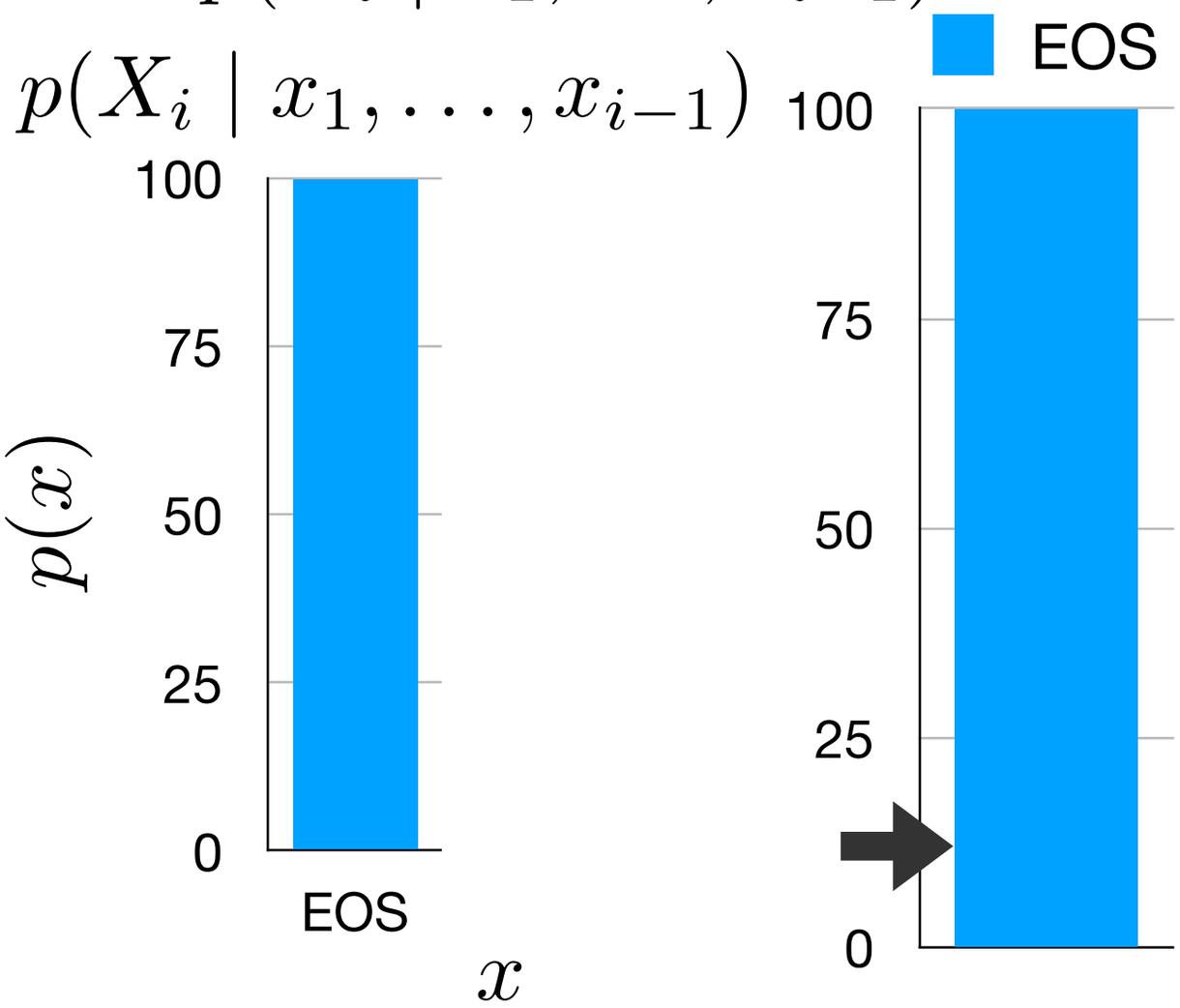
Recap: Ancestral Sampling

$$p(\overline{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

Operation: sample directly from $p(X_i \mid x_1, \ldots, x_{i-1})$

$$\overline{x} = \langle \textit{pun} \, \texttt{EOS} \rangle$$

pun



Adjusting the Temperature

Operation: modify the logits
before computing probabilities

Sneak peek: computing probabilities over wordtypes using pretty much any modern language model

- Score each wordtype independently

$$s(w) = f(w \mid x_1, \ldots, x_{i-1}; \theta)$$ ← logits

- Renormalize using softmax

$$p(X_i = w \mid x_1, \ldots, x_{i-1}) = \frac{\exp(s(w))}{\sum_{w' \in \mathcal{V}} \exp(s(w'))}$$

Adjusting the Temperature

Operation: modify the logits
before computing probabilities

$$s(w) = f(w \mid x_1, \ldots, x_{i-1}; \theta)$$ ← logits

$$p(X_i = w \mid x_1, \ldots, x_{i-1}) = \frac{\exp(s(w))}{\sum_{w' \in \mathcal{V}} \exp(s(w'))}$$

Temperature parameter controls the "smoothness" of this distribution:

$$p(X_i = w \mid x_1, \ldots, x_{i-1}) = \frac{\exp(s(w)/\tau)}{\sum_{w' \in \mathcal{V}} \exp(s(w')/\tau)}$$

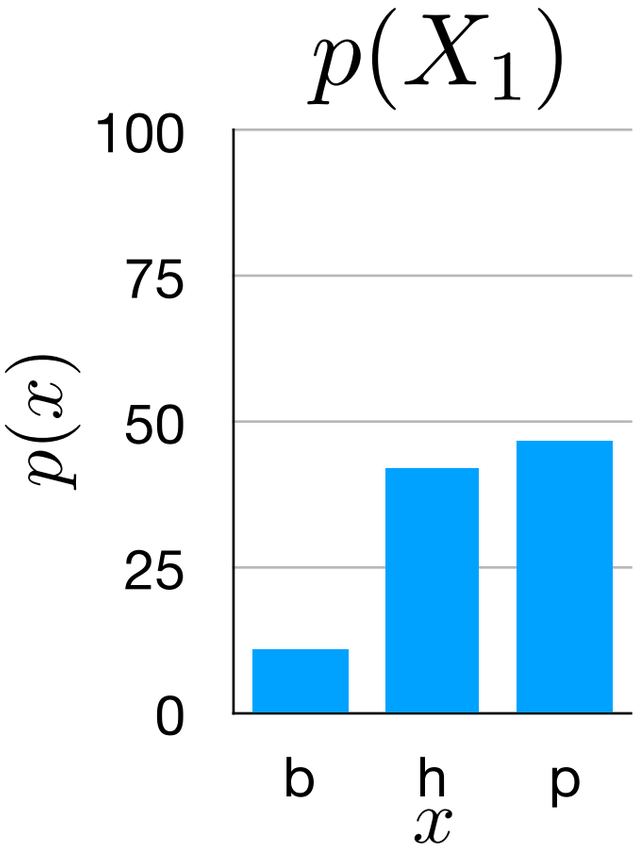
Adjusting the Temperature

Operation: modify the logits
before computing probabilities

$$s(w) = f(w \mid x_1, \ldots, x_{i-1}; \theta)$$ ← logits

$$p(X_i = w \mid x_1, \ldots, x_{i-1}) = \frac{\exp(s(w)/\tau)}{\sum_{w' \in \mathcal{V}} \exp(s(w')/\tau)}$$

- τ = 1: no changes to the probability distribution



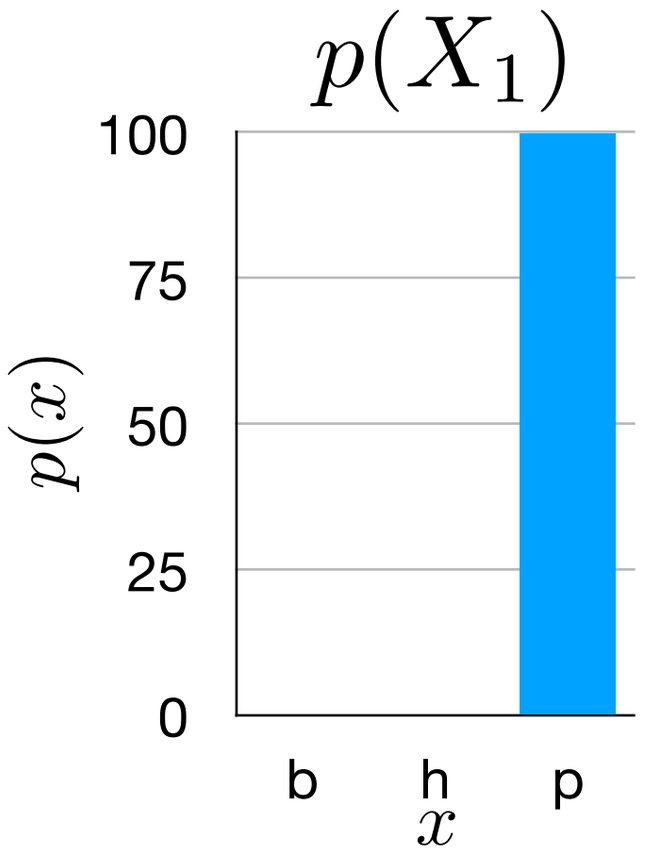
Adjusting the Temperature

Operation: modify the logits *before* computing probabilities

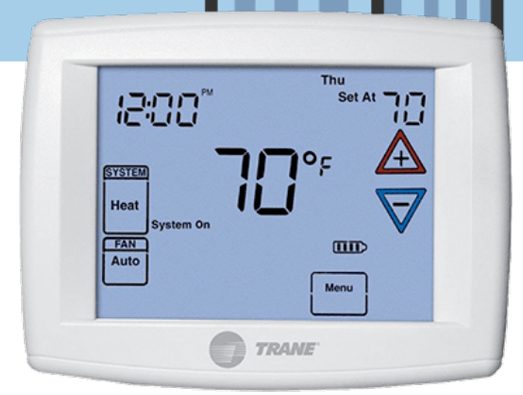
$$s(w) = f(w \mid x_1, \ldots, x_{i-1}; \theta)$$ ← logits

$$p(X_i = w \mid x_1, \ldots, x_{i-1}) = \frac{\exp(s(w)/\tau)}{\sum_{w' \in \mathcal{V}} \exp(s(w')/\tau)}$$

- $\tau = 1$: no changes to the probability distribution
- $\tau \to 0$: relative probability assigned to highest-probability item in distribution increases
  - in practice, setting a temperature of 0 recovers "argmax", putting all of the mass on the highest-probability item



Adjusting the Temperature

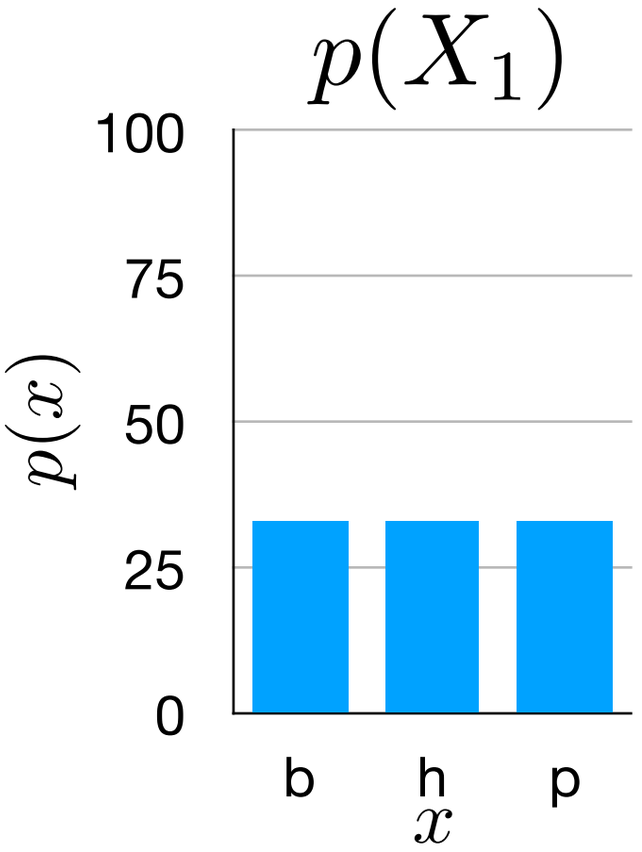


Operation: modify the logits *before* computing probabilities

$$s(w) = f(w \mid x_1, \ldots, x_{i-1}; \theta)$$ ← logits

$$p(X_i = w \mid x_1, \ldots, x_{i-1}) = \frac{\exp(s(w)/\tau)}{\sum_{w' \in \mathcal{V}} \exp(s(w')/\tau)}$$

- τ = 1: no changes to the probability distribution
- τ → 0: relative probability assigned to highest-probability item in distribution increases
- τ → ∞: distribution becomes more and more uniform

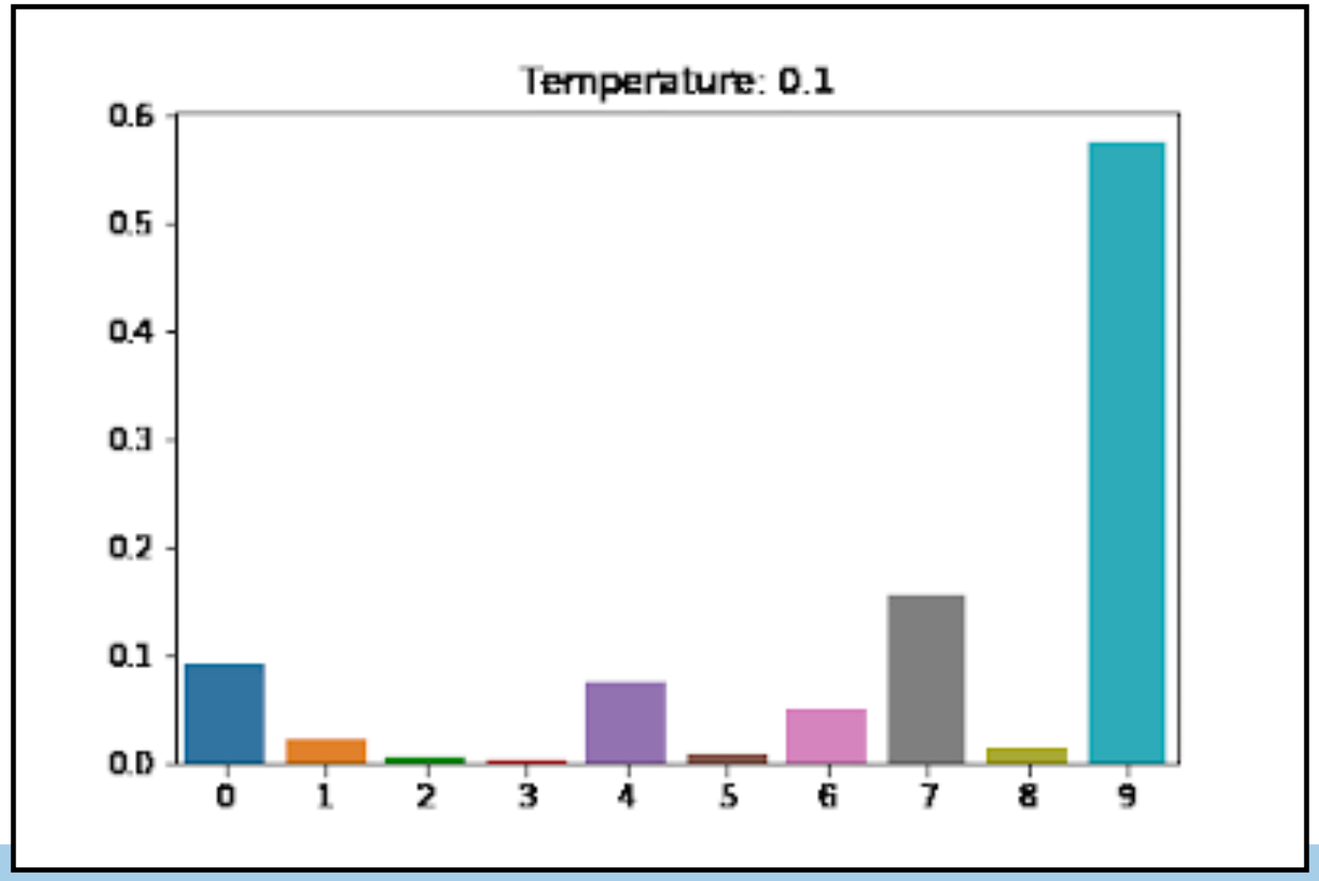


Adjusting the Temperature

Operation: modify the logits
before computing probabilities

$$s(w) = f(w \mid x_1, \ldots, x_{i-1}; \theta)$$ ← logits

$$p(X_i = w \mid x_1, \ldots, x_{i-1}) = \frac{\exp(s(w)/\tau)}{\sum_{w' \in \mathcal{V}} \exp(s(w')/\tau)}$$

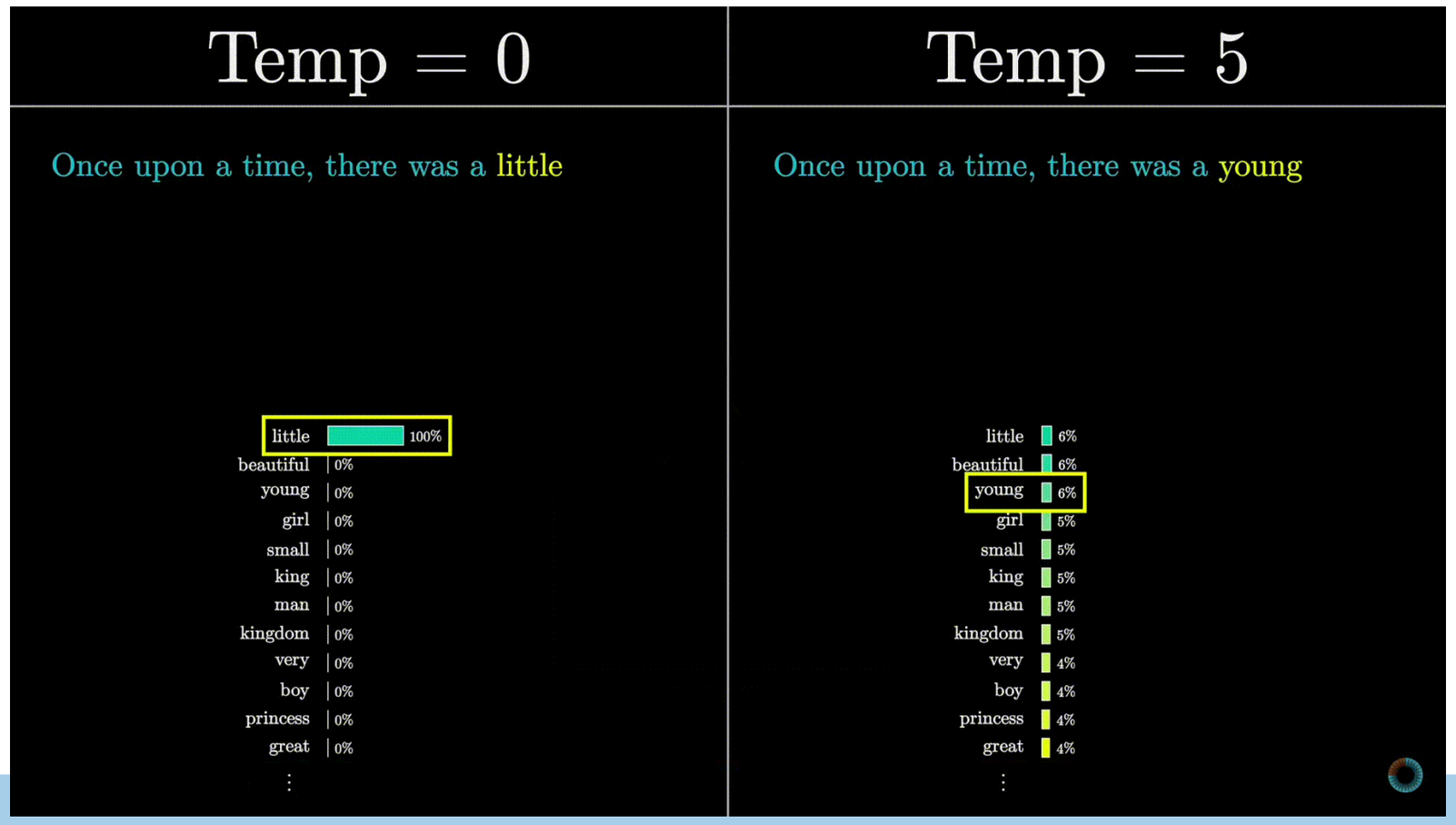


Adjusting the Temperature

Operation: modify the logits
before computing probabilities

$$s(w) = f(w \mid x_1, \ldots, x_{i-1}; \theta)$$ ← logits

$$p(X_i = w \mid x_1, \ldots, x_{i-1}) = \frac{\exp(s(w)/\tau)}{\sum_{w' \in \mathcal{V}} \exp(s(w')/\tau)}$$



Adjusting the Temperature

Operation: modify the logits
before computing probabilities

$$s(w) = f(w \mid x_1, \ldots, x_{i-1}; \theta)$$ ← logits

$$p(X_i = w \mid x_1, \ldots, x_{i-1}) = \frac{\exp(s(w)/\tau)}{\sum_{w' \in \mathcal{V}} \exp(s(w')/\tau)}$$

- **Temperature allows us to control the entropy of the output distribution without changing its relative ranking of items**

- Higher temperature: closer to a uniform distribution

- Lower temperature: “peakier” distribution (in the limit, gives all probability mass to the most probable item)

Finding the Most Probable Sequence

Operation: find $\arg\max_{\overline{x} \in \mathcal{V}^+} p(\overline{x})$

- Why is this hard?

Finding the Most Probable Sequence

Operation: find $\arg\max_{\overline{x} \in \mathcal{V}^+} p(\overline{x})$

- Why is this hard?

- An approximation: greedy “sampling”

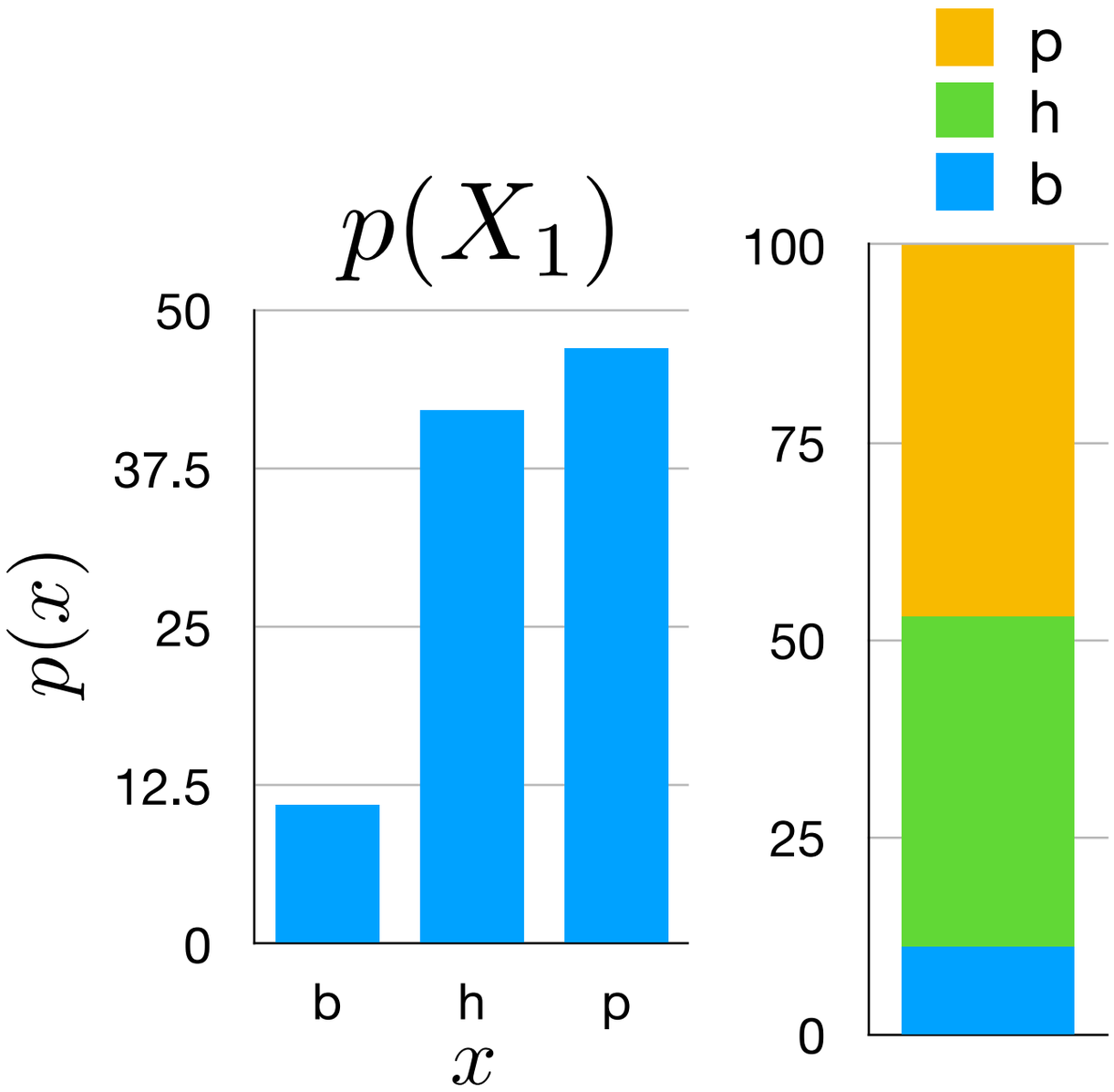
$$x_i \leftarrow \arg\max_{x \in \mathcal{V}} p(X_i \mid x_1, \ldots, x_{i-1})$$

- Just choose the most probable wordtype at each generation step (no random sampling needed)

Finding the Most Probable Sequence

Operation: choose $x_i \leftarrow \arg\max_{x \in \mathcal{V}} p(X_i \mid x_1, \ldots, x_{i-1})$

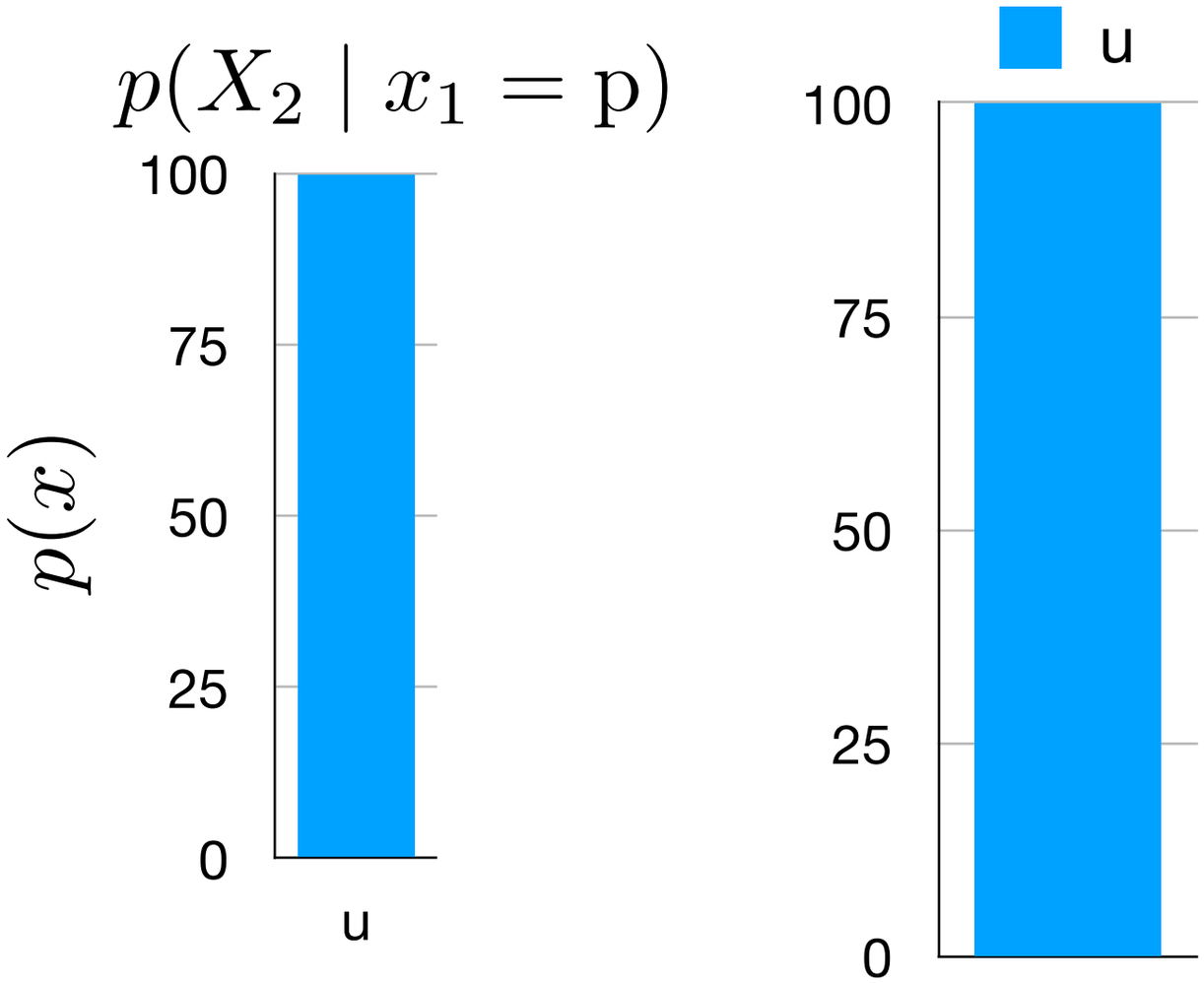
$$\overline{x} = \langle p$$



Finding the Most Probable Sequence

Operation: choose $x_i \leftarrow \arg\max_{x \in \mathcal{V}} p(X_i \mid x_1, \ldots, x_{i-1})$

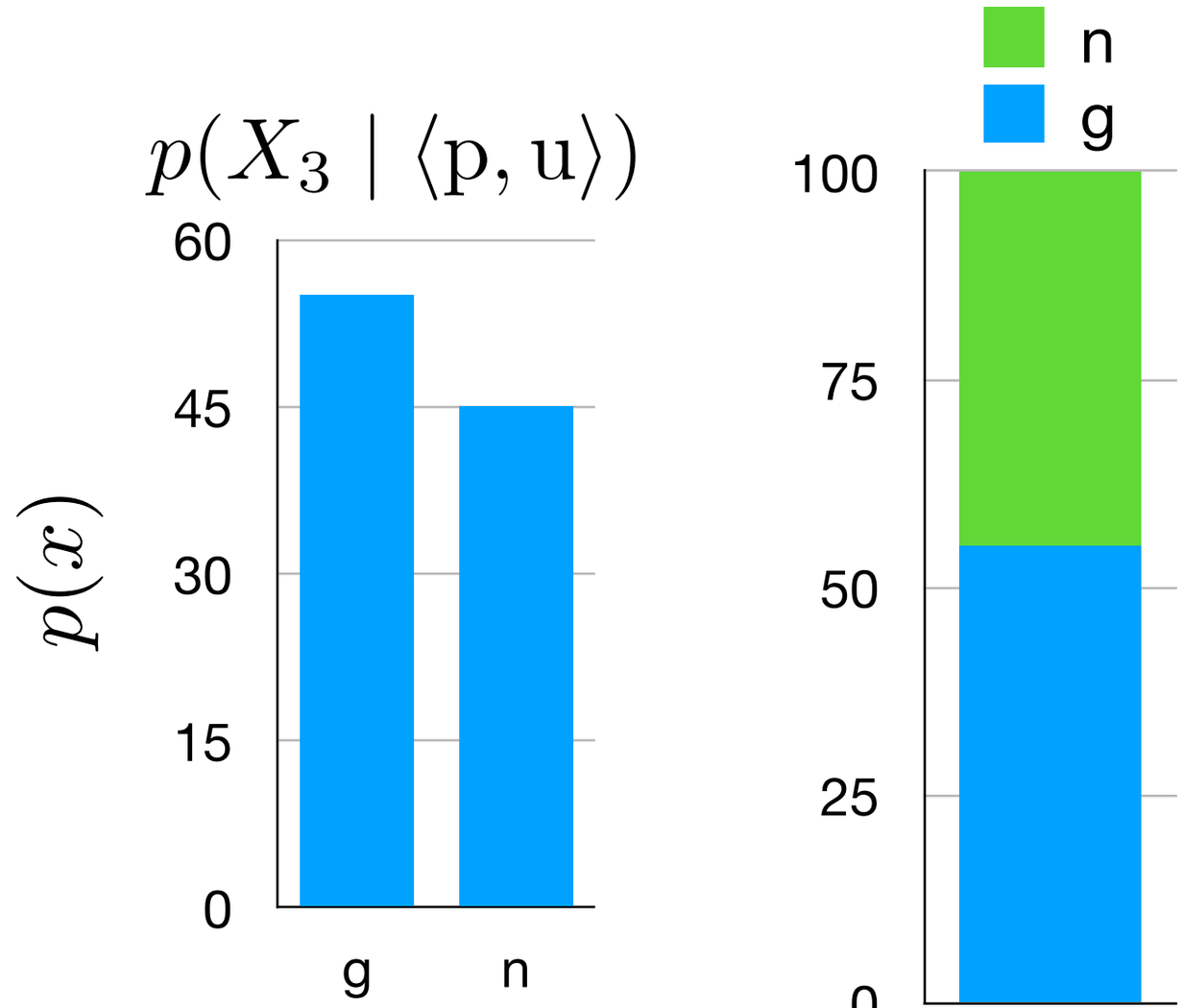
$$\overline{x} = \langle p\, u$$



Finding the Most Probable Sequence

Operation: choose $x_i \leftarrow \arg\max_{x \in \mathcal{V}} p(X_i \mid x_1, \ldots, x_{i-1})$

$$\overline{x} = \langle p\, u\, g$$

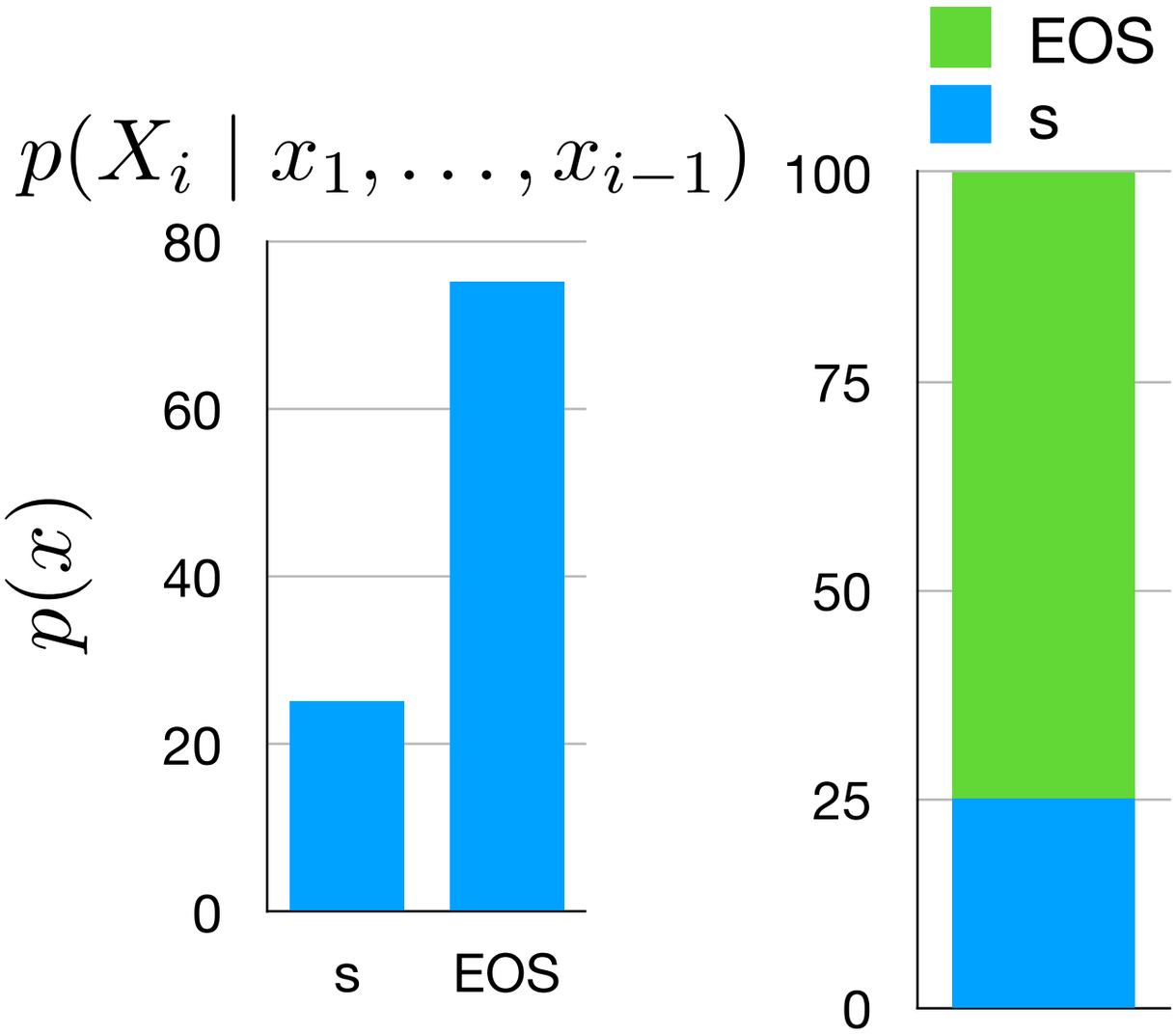


Finding the Most Probable Sequence

Operation: choose $x_i \leftarrow \arg\max_{x \in \mathcal{V}} p(X_i \mid x_1, \ldots, x_{i-1})$

$$\overline{x} = \langle \textit{pug}\,\texttt{EOS} \rangle$$

pug



Finding the Most Probable Sequence

Operation: choose $x_i \leftarrow \arg\max_{x \in \mathcal{V}} p(X_i \mid x_1, \ldots, x_{i-1})$

- Why isn't this guaranteed to get us the highest-probability sequence?

- A better approximation for global argmax: **beam search**
  - During generation, we maintain a "beam" of *n* sequences instead of just one
  - At each generation step *i*,

Finding the Most Probable Sequence

Operation: choose $x_i \leftarrow \arg\max_{x \in \mathcal{V}} p(X_i \mid x_1, \ldots, x_{i-1})$

- Why isn't this guaranteed to get us the highest-probability sequence?

- A better approximation for global argmax: **beam search**
  - During generation, we maintain a "beam" of *n* sequences instead of just one
  - At each generation step *i*,
    - We select the *n* most likely next tokens x_i for each prefix, and create *n* more sequences

Finding the Most Probable Sequence

Operation: choose $x_i \leftarrow \arg\max_{x \in \mathcal{V}} p(X_i \mid x_1, \ldots, x_{i-1})$

- Why isn't this guaranteed to get us the highest-probability sequence?

- A better approximation for global argmax: **beam search**
  - During generation, we maintain a "beam" of *n* sequences instead of just one
  - At each generation step *i*,
    - We select the *n* most likely next tokens x_i for each prefix, and create *n* more sequences
    - Then we look at all the n^2 sequences so far, and discard all but the *n* most likely sequences

Finding the Most Probable Sequence

Operation: choose $x_i \leftarrow \arg\max_{x \in \mathcal{V}} p(X_i \mid x_1, \ldots, x_{i-1})$

- Why isn't this guaranteed to get us the highest-probability sequence?

- A better approximation for global argmax: **beam search**
  - During generation, we maintain a "beam" of *n* sequences instead of just one
  - At each generation step *i*,
    - We select the *n* most likely next tokens x_i for each prefix, and create *n* more sequences
    - Then we look at all the n^2 sequences so far, and discard all but the *n* most likely sequences
  - At the end, we select the sequence that has the highest probability among the set

Beam Search, $n = 3$

$p(X_1)$

the

0.4

in

0.3

and

0.2

every

0.1

way

0.1

Beam Search, $n = 3$

$p(X_1)$

the

0.4

in

0.3

and

0.2

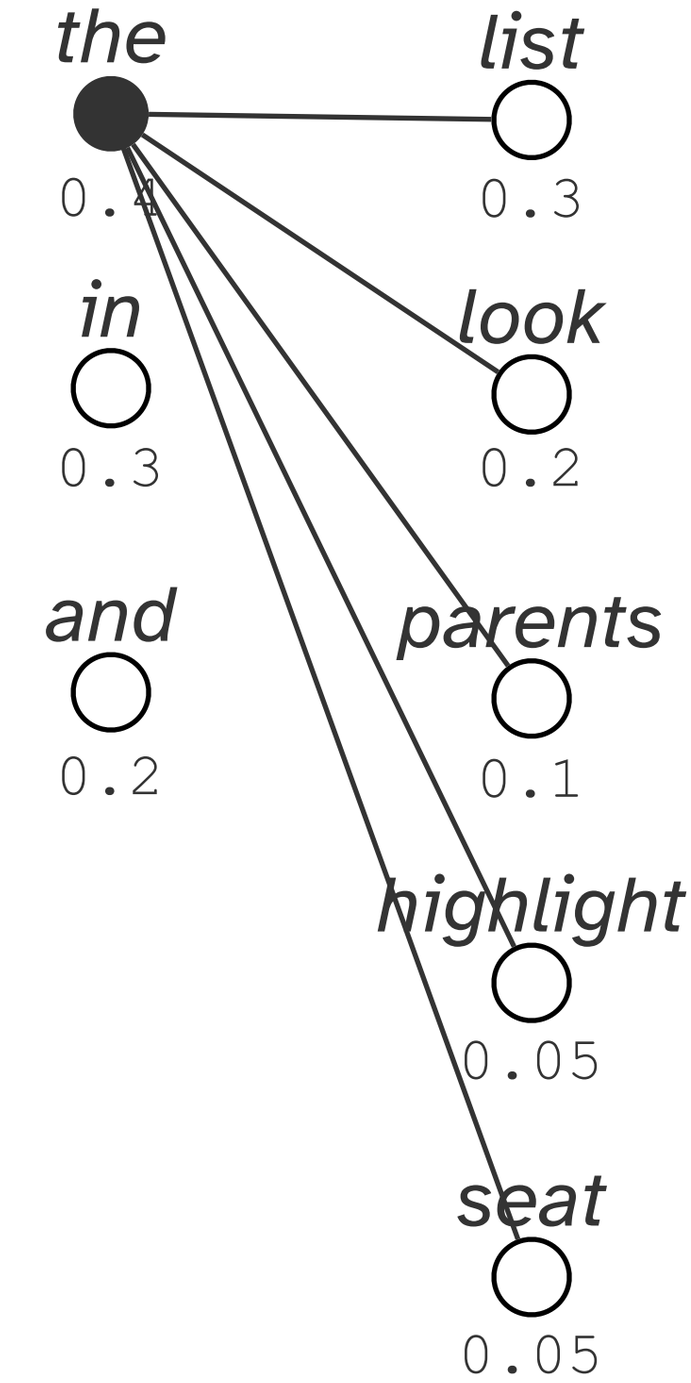
Discard all but the top 3 continuations

Beam

Prefix	Probability
the	0.4
in	0.3
and	0.2

Beam Search, $n = 3$

$p(X_1)\ p(X_2 \mid x_1 = \text{the})$



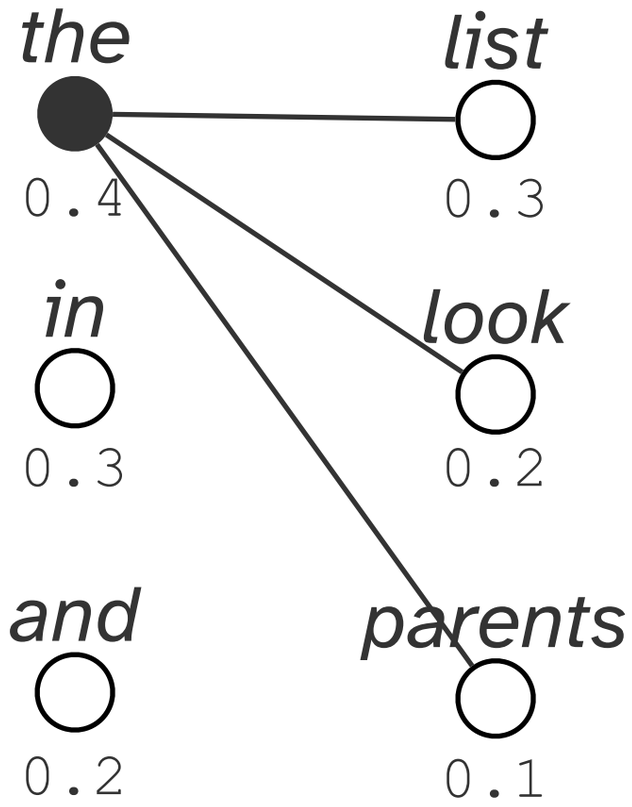
Continue with beam item #1 as prefix

Beam

Prefix	Probability
the	0.04
in	0.02
and	0.02

Beam Search, $n = 3$

$$p(X_1)\ p(X_2 \mid x_1 = \text{the})$$



Discard all but the top 3 continuations

Current Beam Candidates

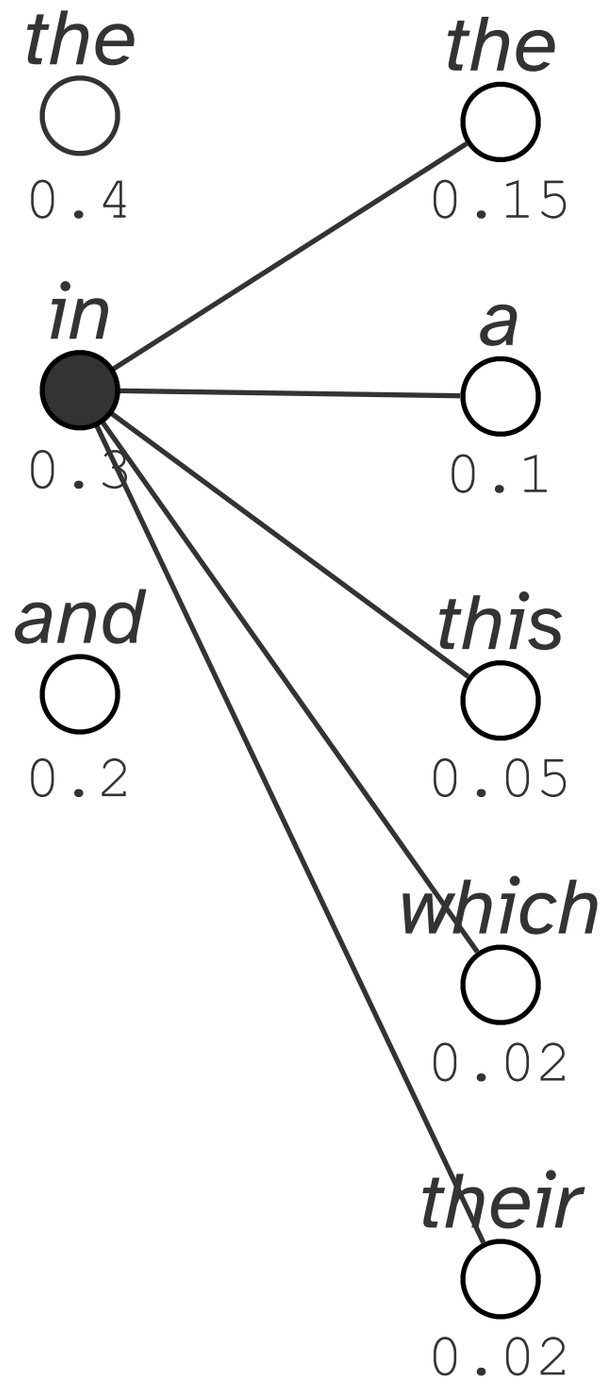
Prefix	Probability
the list	0.4*0.3
the look	0.3*0.2
the parents	0.2*0.1

Beam

Prefix	Probability
the	0.04
in	0.02
and	0.02

Beam Search, $n = 3$

$$p(X_1)p(X_2 \mid x_1 = \text{in})$$



Continue with beam item #2 as prefix

Current Beam Candidates

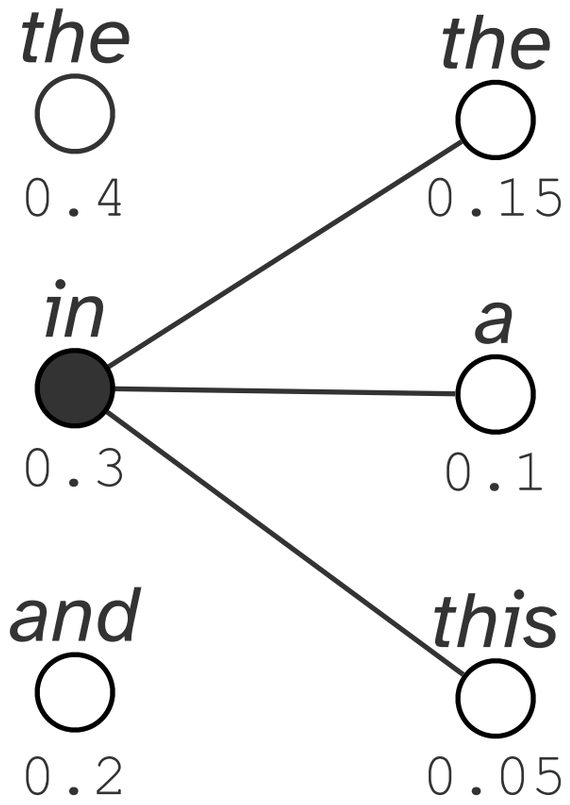
Prefix	Probability
the list	0.4*0.3
the look	0.3*0.2
the parents	0.2*0.1

Beam

Prefix	Probability
the	0.04
in	0.02
and	0.02

Beam Search, $n = 3$

$$p(X_1)p(X_2 \mid x_1 = \text{in})$$



Discard all but the top 3 continuations

Current Beam Candidates

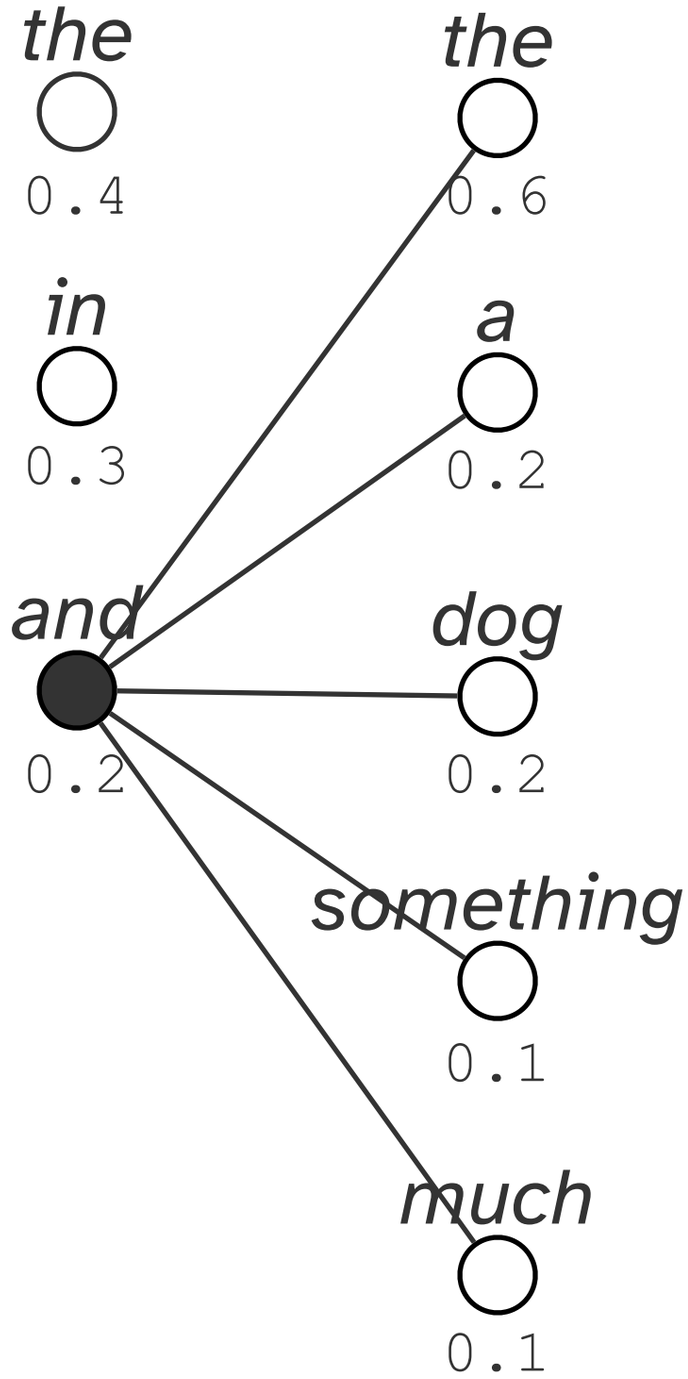
Prefix	Probability
the list	0.4*0.3
the look	0.3*0.2
the parents	0.2*0.1
in the	0.3*0.15
in a	0.3*0.1
in this	0.3*0.05

Beam

Prefix	Probability
the	0.04
in	0.02
and	0.02

Beam Search, $n = 3$

$$p(X_1)\, p(X_2 \mid x_1 = \text{and})$$



Continue with beam item #3 as prefix

Current Beam Candidates

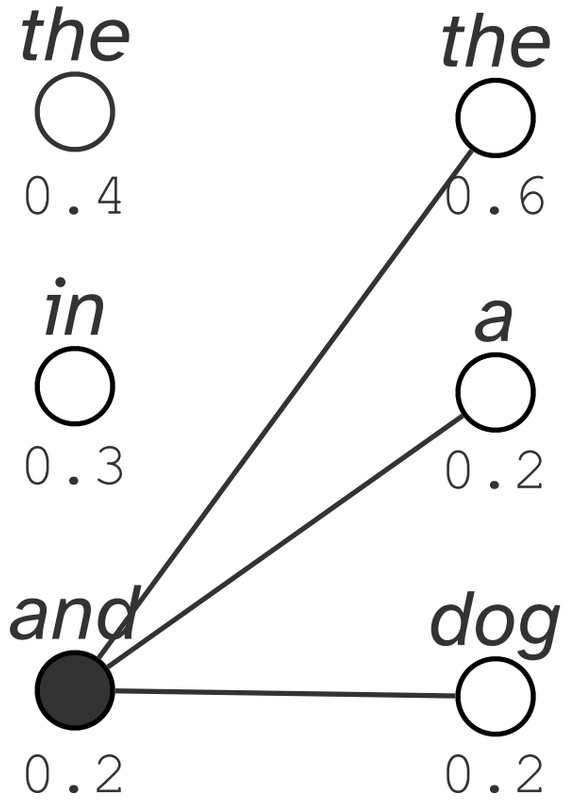
Prefix	Probability
the list	0.4*0.3
the look	0.3*0.2
the parents	0.2*0.1
in the	0.3*0.15
in a	0.3*0.1
in this	0.3*0.05

Beam

Prefix	Probability
the	0.04
in	0.02
and	0.02

Beam Search, $n = 3$

$$p(X_1)\, p(X_2 \mid x_1 = \text{and})$$



Discard all but the top 3 continuations

Current Beam Candidates

Prefix	Probability
the list	0.4*0.3
the look	0.3*0.2
the parents	0.2*0.1
in the	0.3*0.15
in a	0.3*0.1
in this	0.3*0.05
and the	0.2*0.6
and a	0.2*0.2
and dog	0.2*0.2

Beam

Prefix	Probability
the	0.04
in	0.02
and	0.02

Beam Search, $n = 3$

Compute probabilities
of all candidates

Current Beam Candidates

Prefix	Probability
the list	0.12
the look	0.06
the parents	0.02
in the	0.05
in a	0.03
in this	0.02
and the	0.12
and a	0.04
and dog	0.04

Beam

Prefix	Probability
the	0.04
in	0.02
and	0.02

Beam Search, $n = 3$

Refresh the beam
with top n candidates

Current Beam Candidates

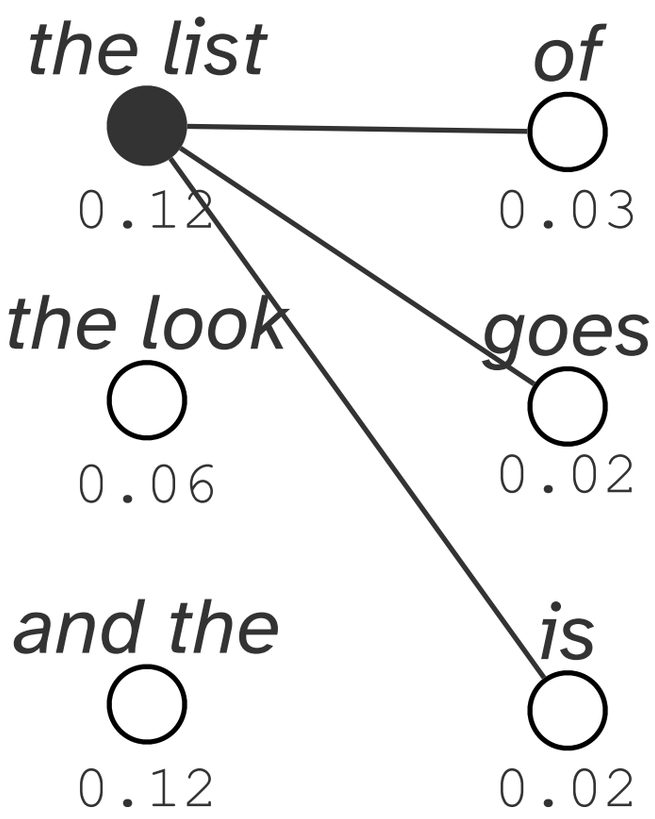
Prefix	Probability
the list	0.12
the look	0.06
the parents	0.02
in the	0.05
in a	0.03
in this	0.02
and the	**0.12**
and a	0.04
and dog	0.04

Beam

Prefix	Probability
the list	0.12
the look	0.06
and the	0.12

Beam Search, $n = 3$

$p(\overline{x})$ $p(X_3 \mid \overline{x} = \text{the list})$



Keep generating with new beam

Current Beam Candidates

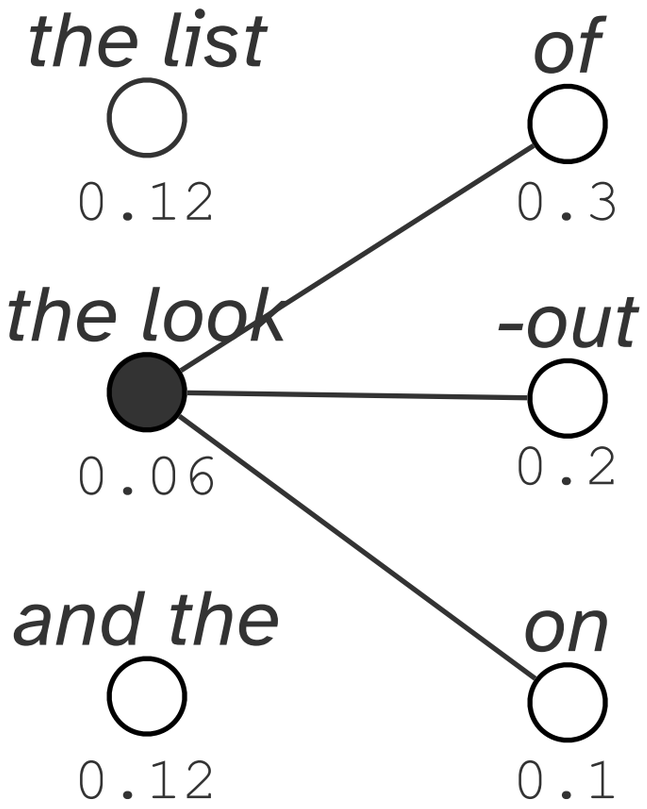
Prefix	Probability
the list of	0.004
the list goes	0.002
the list is	0.002

Beam

Prefix	Probability
the list	0.12
the look	0.06
and the	0.12

Beam Search, $n = 3$

$p(\overline{x})$ $p(X_3 \mid \overline{x} = \text{the look})$



Keep generating with new beam

Current Beam Candidates

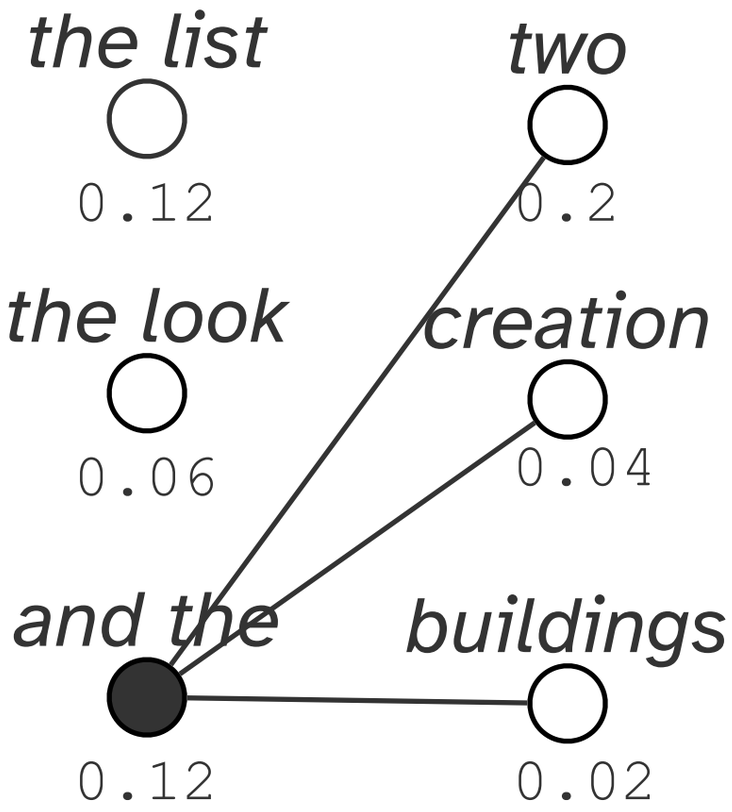
Prefix	Probability
the list of	0.004
the list goes	0.002
the list is	0.002
the look of	0.018
the lookout	0.012
the look on	0.006

Beam

Prefix	Probability
the list	0.12
the look	0.06
and the	0.12

Beam Search, $n = 3$

$p(\overline{x})$ $p(X_3 \mid \overline{x} = \text{the look})$



Keep generating with new beam

Current Beam Candidates

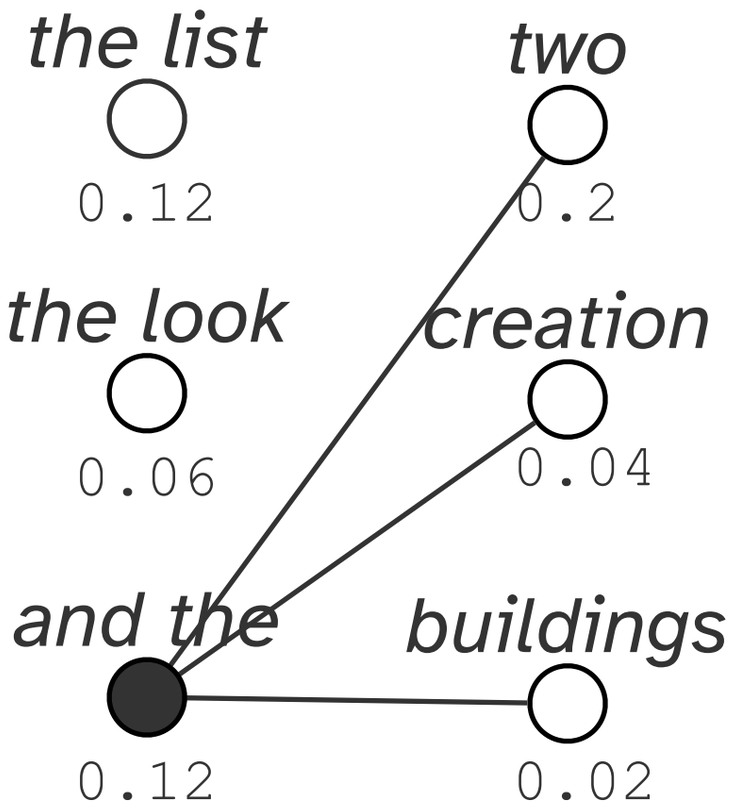
Prefix	Probability
the list of	0.004
the list goes	0.002
the list is	0.002
the look of	0.018
the lookout	0.012
the look on	0.006
and the two	0.024
and the creation	0.005
and the buildings	0.002

Beam

Prefix	Probability
the list	0.12
the look	0.06
and the	0.12

Beam Search, $n = 3$

$p(\overline{x})$ $p(X_3 \mid \overline{x} = \text{the look})$



Refresh the beam with top *n* candidates

Current Beam Candidates

Prefix	Probability
the list of	0.004
the list goes	0.002
the list is	0.002
the look of	**0.018**
the lookout	**0.012**
the look on	0.006
and the two	**0.024**
and the creation	0.005
and the buildings	0.002

Beam

Prefix	Probability
the look of	0.018
the lookout	0.012
and the two	0.024

Beam Search, $n = 3$

- How do we know when to stop?
  - When all of the items in the beam have EOS (we don't expand these prefixes, just keep them around for the end)
  - Or, when we've reached a maximum sequence length
- Let's say we're done sampling at this point
- We'll select the sequence with the highest probability in the beam

Beam

Prefix	Probability
the look of	0.018
the lookout	0.012
and the two	0.024

Beam Search, $n = 3$

- How do we know when to stop?
  - When all of the items in the beam have EOS (we don't expand these prefixes, just keep them around for the end)
  - Or, when we've reached a maximum sequence length
- Let's say we're done sampling at this point
- We'll select the sequence with the highest probability in the beam

Beam

Prefix	Probability
the look of	0.018
the lookout	0.012
and the two	**0.024**

Beam Search, $n = 3$

- How do we know when to stop?
  - When all of the items in the beam have EOS (we don't expand these prefixes, just keep them around for the end)
  - Or, when we've reached a maximum sequence length
- Let's say we're done sampling at this point
- We'll select the sequence with the highest probability in the beam
- What if our sequences have different lengths?

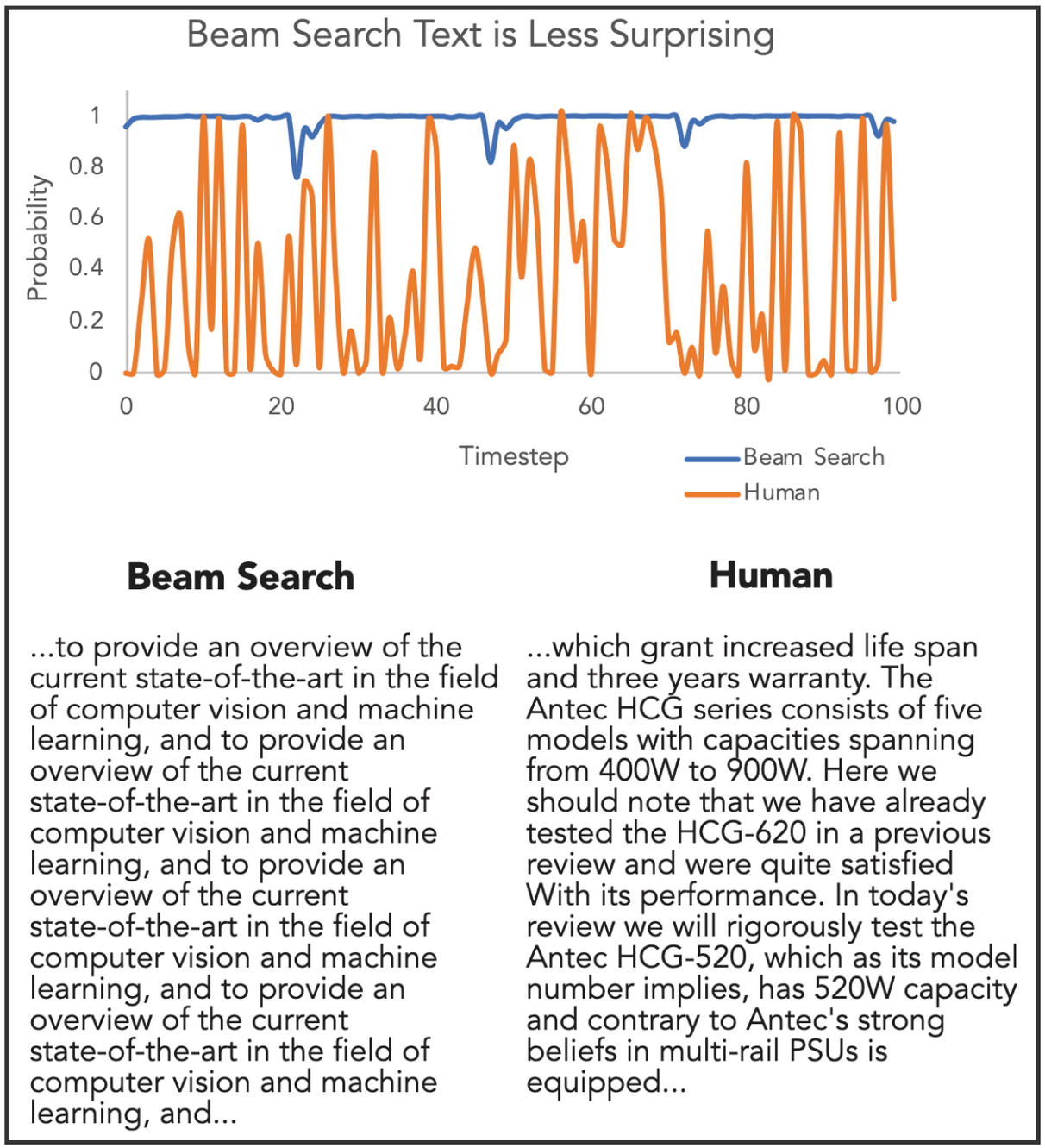
Length normalization: $$p(\overline{x}) \propto -\frac{1}{|\overline{x}|^{\alpha}} \sum_{i=1}^{|\overline{x}|} \log p(x_i \mid x_1, \ldots, x_{i-1})$$

Masking Out Wordtypes

- Should we always try to approximate the argmax?

Masking Out Wordtypes

- Should we always try to approximate the argmax?
- Maybe not!
- Argmax produces repetitive, less diverse, and overall *too*-probable output sequences
- What's missing?



Holtzman et al. 2019

Masking Out Wordtypes

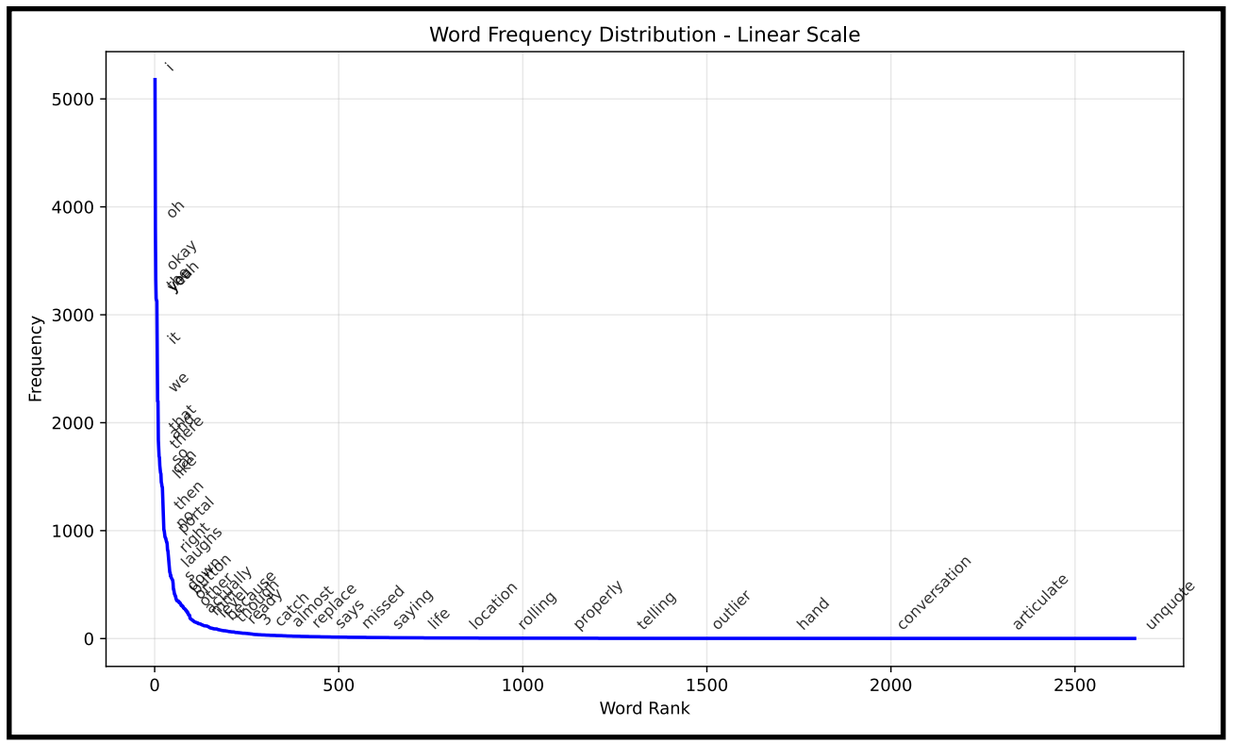
- Should we always try to approximate the argmax?

- Maybe not!

- Argmax produces repetitive, less diverse, and overall *too*-probable output sequences

- What's missing?

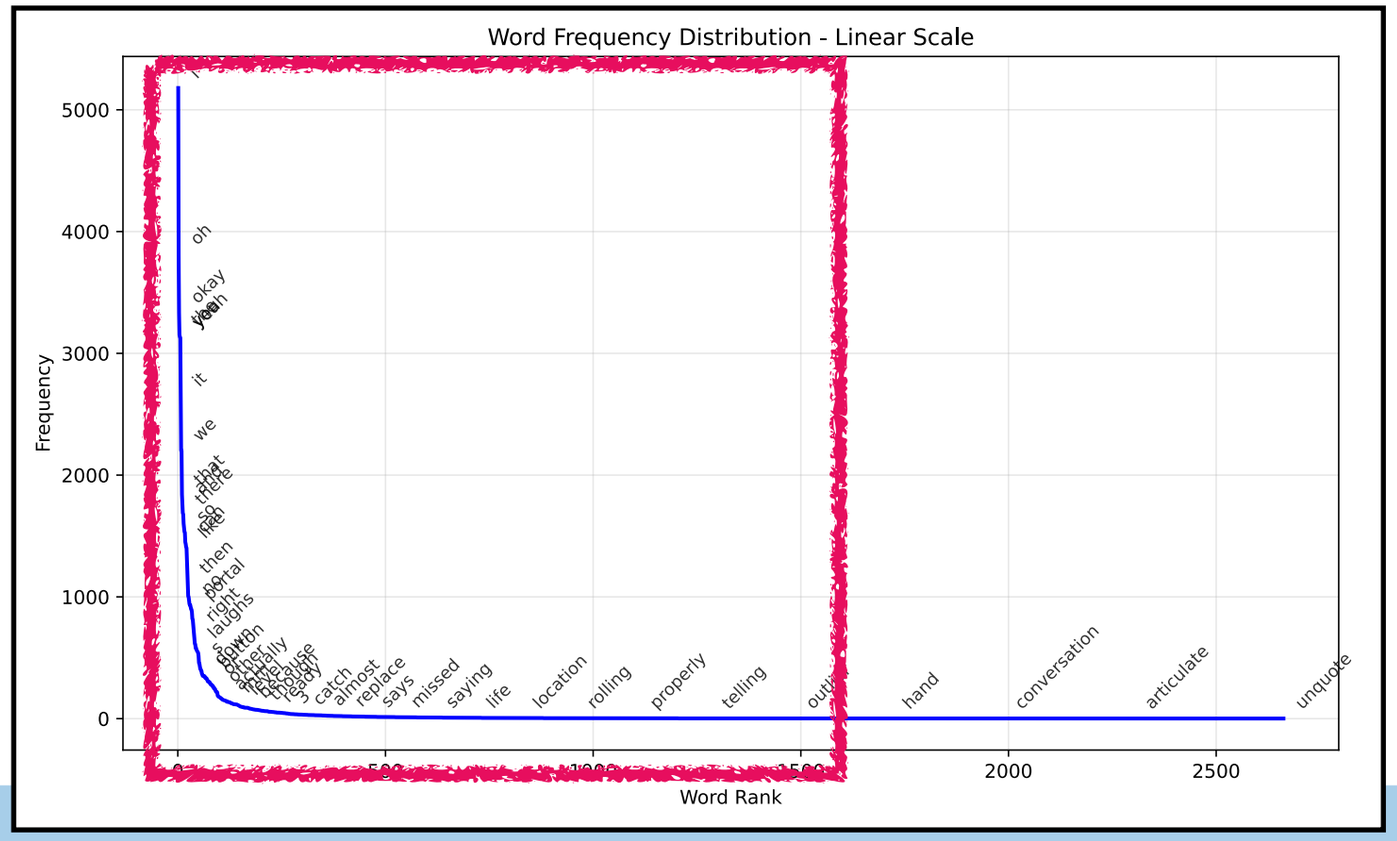
- Long tail!



Masking Out Wordtypes

Operation: ϵ sampling

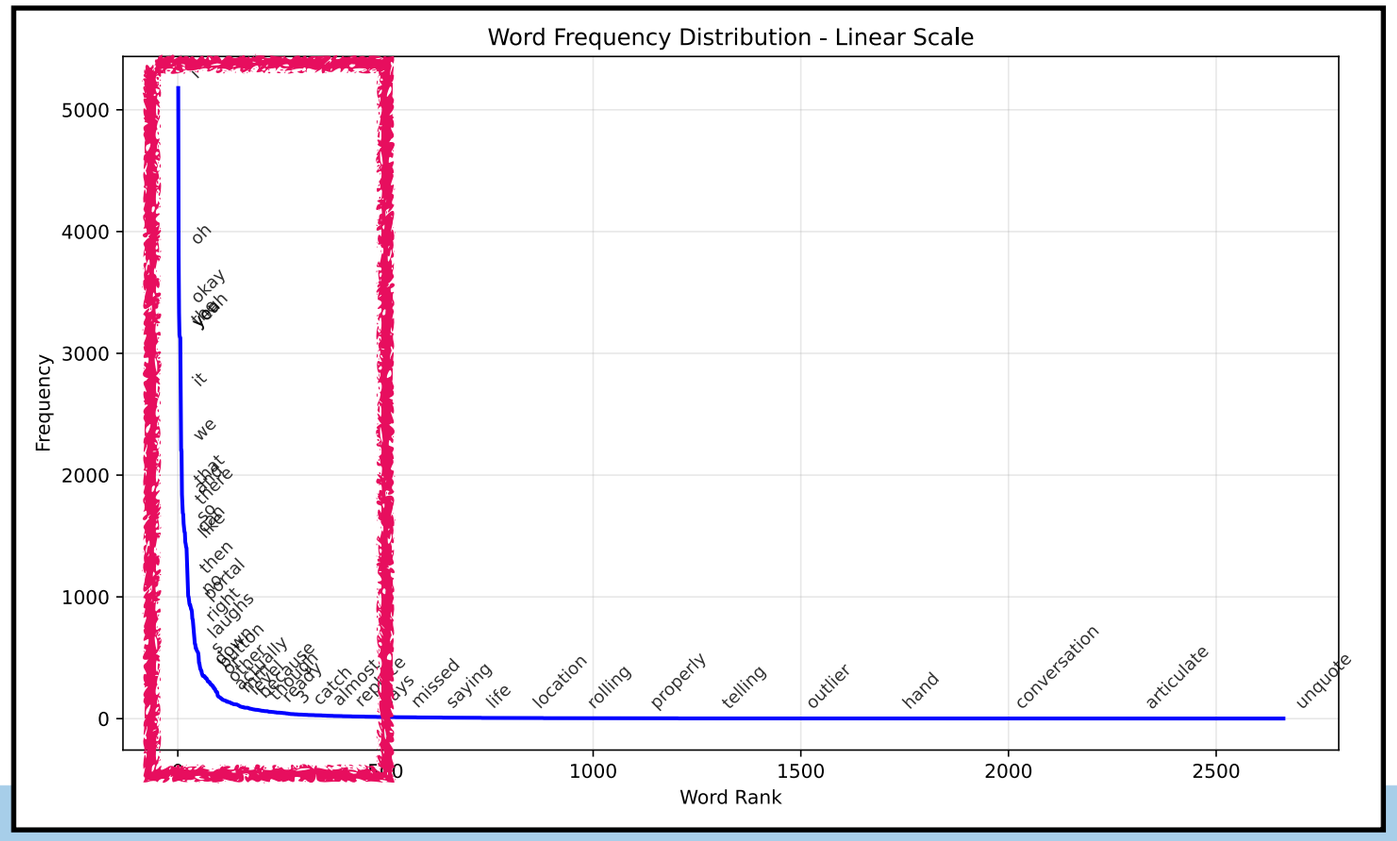
- Identify the set of *n* tokens $\mathcal{E}$ such that $\forall\ x \in \mathcal{E}$, $p(x \mid x_1, \ldots, x_{i-1}) \geq \epsilon$
- Set the probabilities of all but these tokens to 0
- Renormalize by dividing remaining probabilities by $\sum_{x \in \mathcal{E}} p(X_i \mid x_1, \ldots, x_{i-1})$



Masking Out Wordtypes

Operation: top-k sampling

- Identify the set of *k* tokens $\mathcal{K}$ that have the highest probabilities under $p(X_i \mid x_1, \ldots, x_{i-1})$
- Set the probabilities of all but these tokens to 0
- Renormalize by dividing remaining probabilities by $\sum_{x \in \mathcal{K}} p(X_i \mid x_1, \ldots, x_{i-1})$

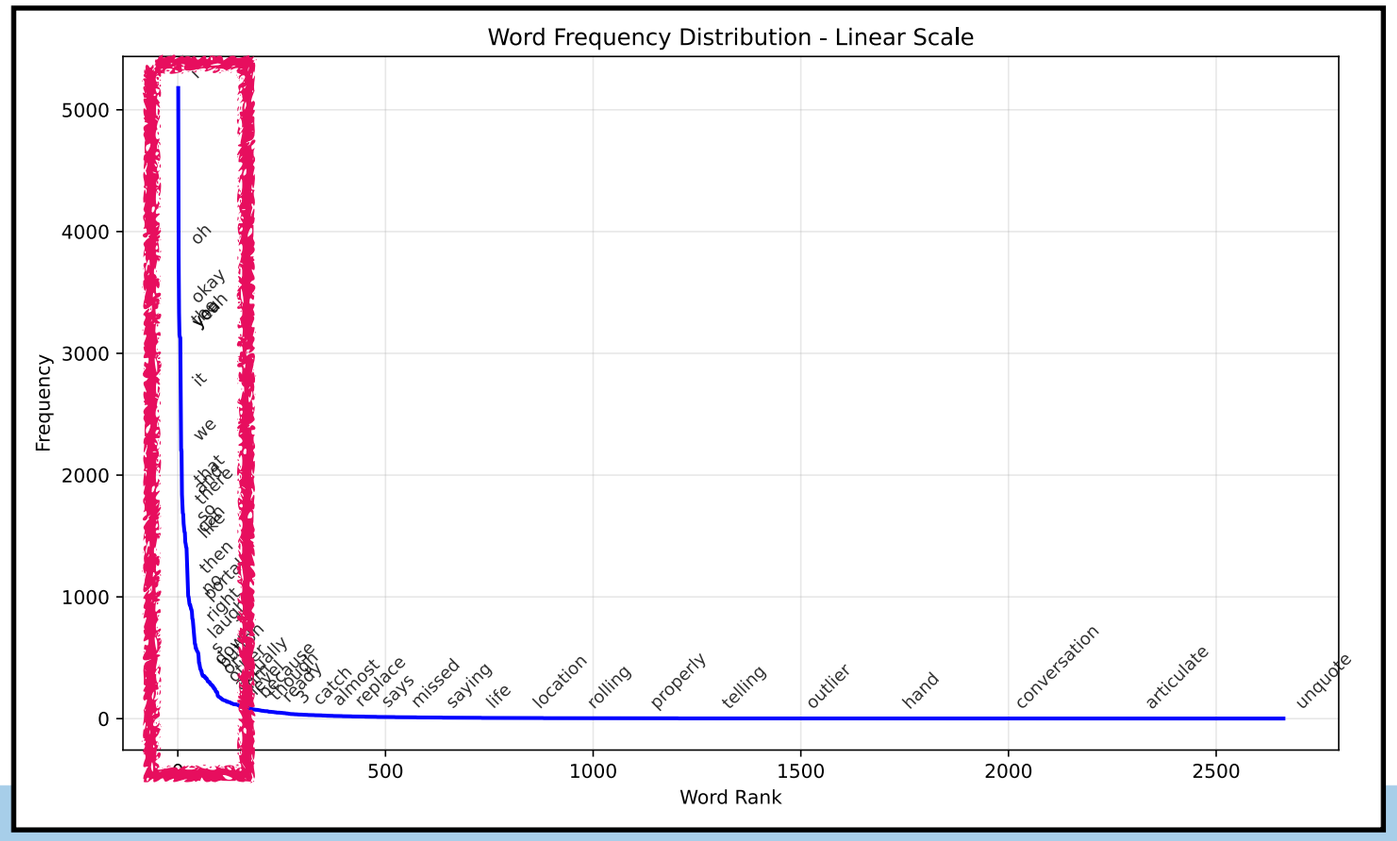


Masking Out Wordtypes

Operation: top-p (nucleus) sampling

- Identify the set of *n* tokens $\mathcal{P}$ that have the highest probabilities under $p(X_i \mid x_1, \ldots, x_{i-1})$ and their cumulative probability is *p*
- Set the probabilities of all but these tokens to 0
- Renormalize by dividing remaining probabilities by

$$\sum_{x \in \mathcal{P}} p(X_i \mid x_1, \ldots, x_{i-1}) = p$$



Masking Out Wordtypes

Operation: constrained decoding

- For some tasks, we have additional information about what wordtypes can or cannot be next
- E.g., in code generation, I can't generate more) than I have (
- While modern LLMs can learn these patterns from data at scale, it can sometimes still be useful to constrain our output space

Masking Out Wordtypes

Operation: constrained decoding

- For some tasks, we have additional information about what wordtypes can or cannot be next
- E.g., in code generation, I can't generate more) than I have (
- While modern LLMs can learn these patterns from data at scale, it can sometimes still be useful to constrain our output space
- Similar to before: given a set of possible continuations $\mathcal{C} \subseteq \mathcal{V}$ we will set the probabilities of all other tokens to 0, then renormalize using $\sum_{x \in \mathcal{C}} p(X_i \mid x_1, \ldots, x_{i-1})$