

CS11-711 Advanced NLP

Tokenization and Decoding Algorithms

Daphne Ippolito and Fernando Diaz



<https://cmu-anlp.org>

<https://github.com/cmu-l3/anlp-spring2026-code>

Slides adapted from: Matthew Finlayson, Amanda Bertsch, and Sean Welleck

Outline

- Recap
- Generating text with a language model
- Speeding up generation
- Advanced tokenization methods

Recap

Recap - Transformer model architectures

- Decoder-only?
- Encoder-only?
- Encoder-decoder?

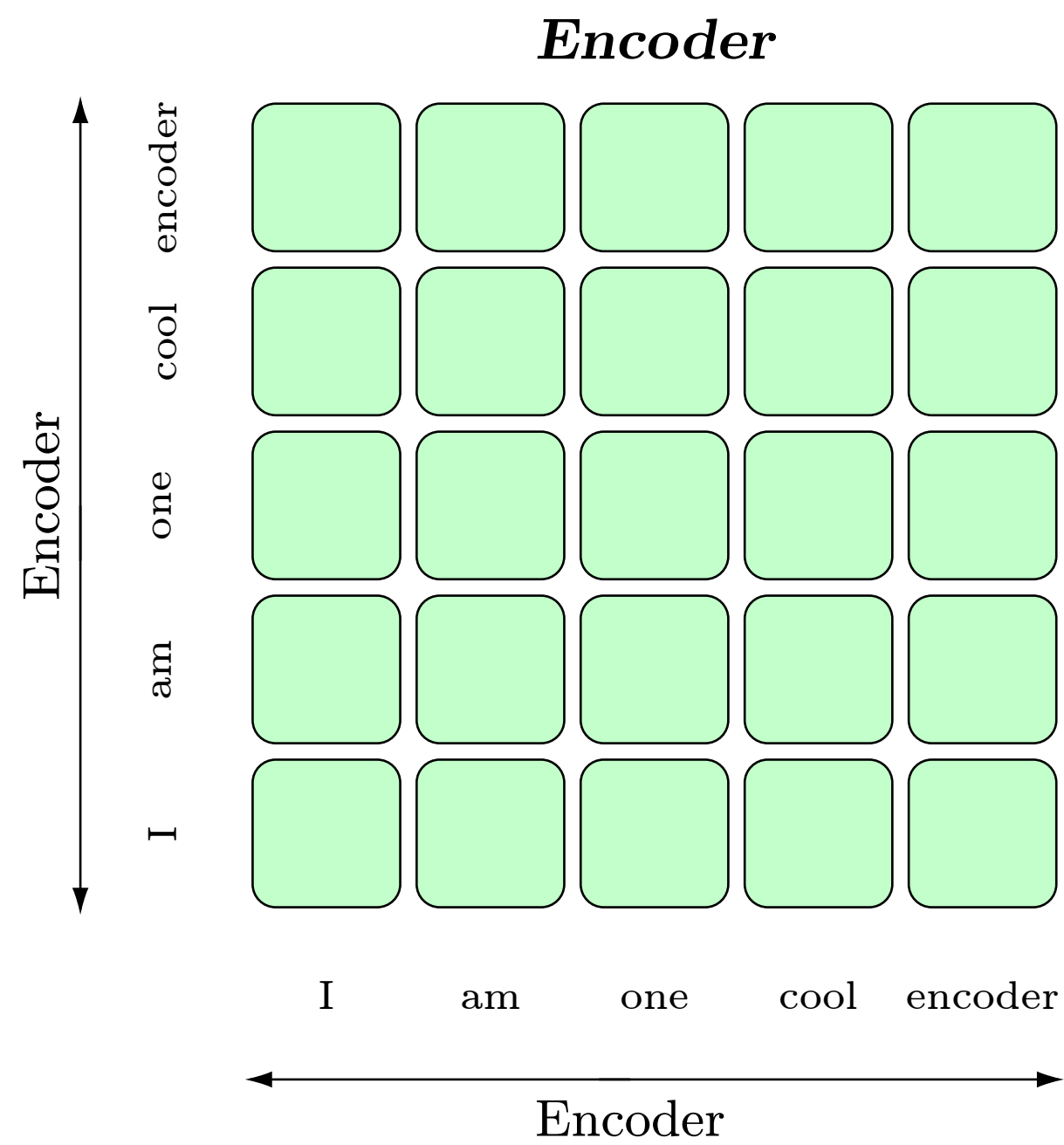
Recap - Transformer model architectures

- Decoder-only?
- Encoder-only?
- Encoder-decoder?

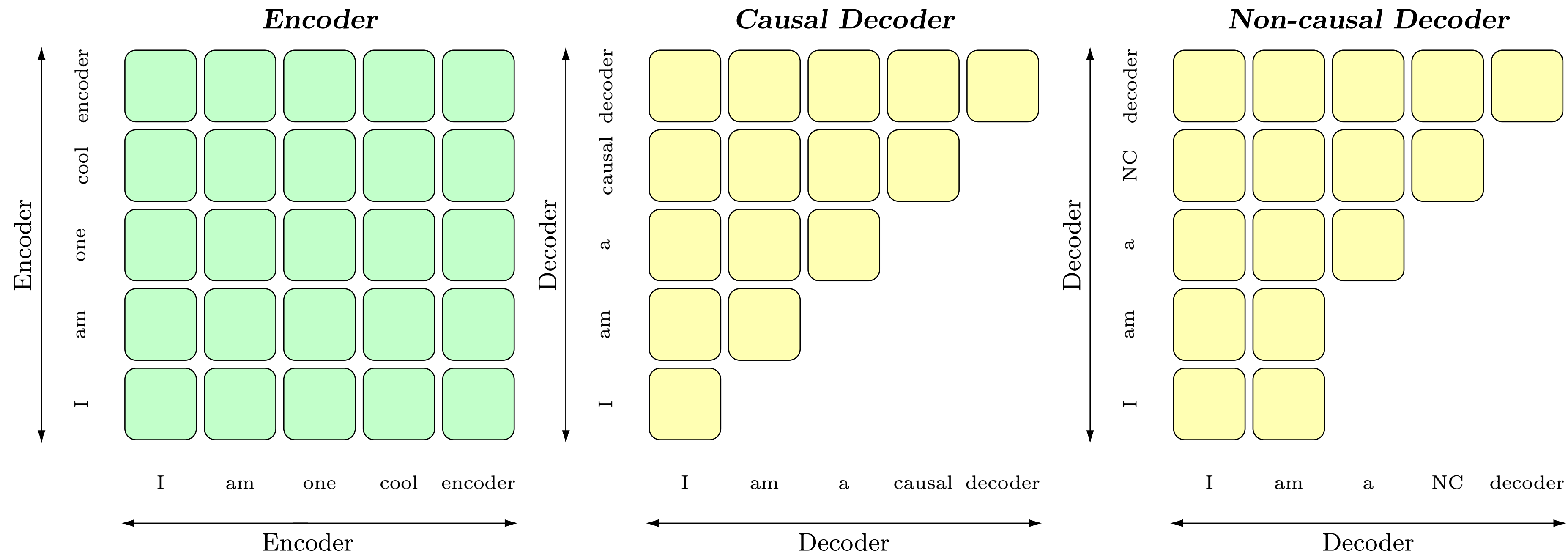
Key differences:

- Attention masking mechanism
- Available choices for loss function

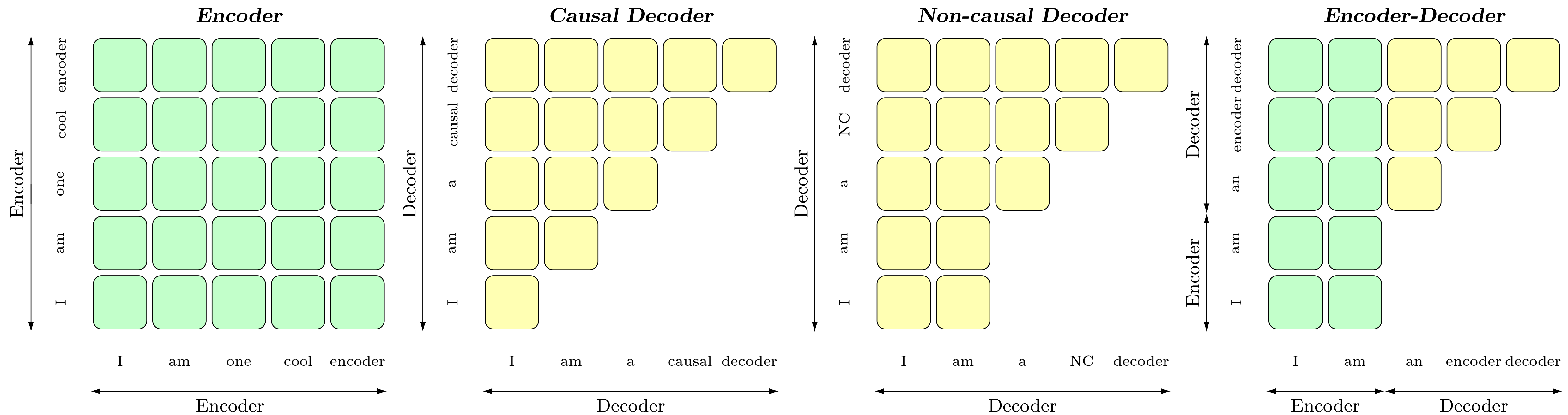
Recap - Transformer model architectures



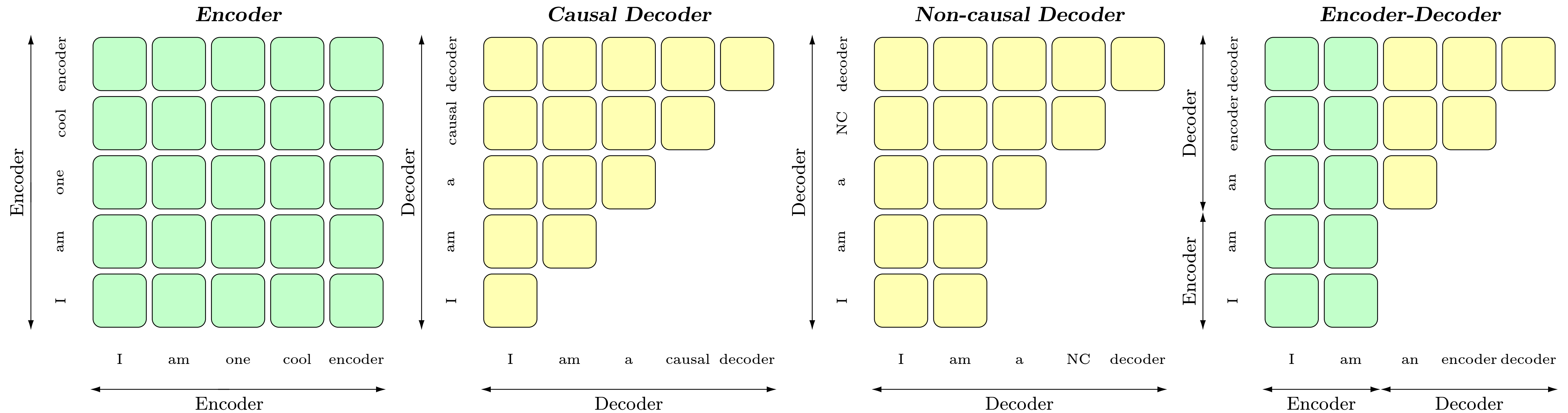
Recap - Transformer model architectures



Recap - Transformer model architectures



Recap - Transformer model architectures



Loss options:

- Per-position classification
- Sequence classification

Loss options:

- Next-token prediction

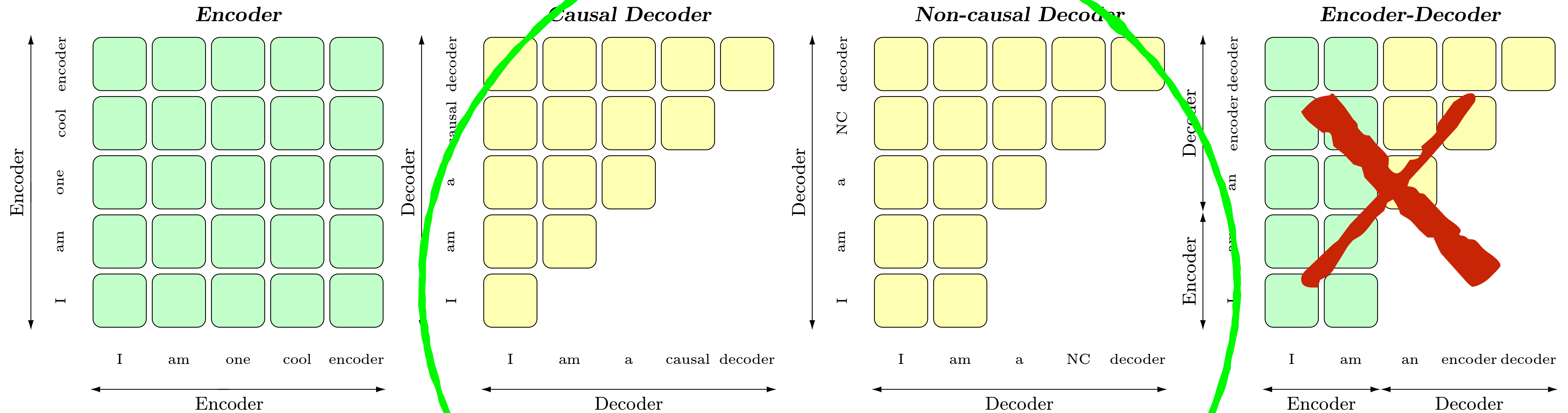
Loss options:

- Next-token prediction on causal part of sequence

Loss options:

- Next-token prediction on decoder sequence
 - e.g. masked language modeling

Recap - Transformer model architectures



Loss options:

- Per-position classification
- Sequence classification

Loss options:

- Next-token prediction

Loss options:

- Next-token prediction on causal part of sequence

Loss options:

- Next-token prediction on decoder sequence
- e.g. masked language modeling

True-or-false: Multi-head attention with $d_k = d_v = d_m/h$ and $h > 1$ heads results in more trainable weights than single-head attention.

True-or-false: Multi-head attention with $d_k = d_v = d_m/h$ and $h > 1$ heads results in more trainable weights than single-head attention.

A. False. They result in the same number of weights.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

$$W_i^Q \in \mathbb{R}^{d_m \times d_k}, W_i^K \in \mathbb{R}^{d_m \times d_k}, W_i^V \in \mathbb{R}^{d_m \times d_v}, W^O \in \mathbb{R}^{hd_v \times d_m}.$$

This leads to $2d_md_k + (1 + h)d_vd_m$ parameters.

In contrast:

$$\text{SingleHead}(Q, K, V) = \text{Attention}(QW^Q, KW^K, VW^V)W^O$$
$$W_i^Q \in \mathbb{R}^{d_m \times d_k}, W_i^K \in \mathbb{R}^{d_m \times d_k}, W_i^V \in \mathbb{R}^{d_m \times d_v}, W^O \in \mathbb{R}^{d_m \times d_m}$$

Since $hd_v = d_m$, these have the exact same number of learnable weights.

What would happen if you set $d_k = d_v > d_m/h$?

- A. It's not possible to do this. The dimensions for the matrix multiplications don't work out.
- B. The total number of trainable parameters increases.
- C. The total number of trainable parameters decreases.

What would happen if you set $d_k = d_v > d_m/h$?

- A. It's not possible to do this. The dimensions for the matrix multiplications don't work out.
- B. The total number of trainable parameters increases.
- C. The total number of trainable parameters decreases.

Why not A? You can set W^O to whatever it needs to be to project from the concatenation of the different heads back down to the model dimension d_m .

Which of the following is an improvement over grouped-query attention?

- A. It's not possible to do this. The dimensions for the matrix multiplications don't work out.
- B. The total number of trainable parameters increases.
- C. The total number of trainable parameters decreases.

Which of the following is an improvement over grouped-query attention?

- A. It's not possible to do this. The dimensions for the matrix multiplications don't work out.
- B. ☒ The total number of trainable parameters increases.
- C. The total number of trainable parameters decreases.

How do I decode^[1] text using a decoder^[2]-only LM? Did I implement my tokenizer's decode^[3] function correctly?

What's the meaning of decode^[1]?

What's the meaning of decode^[2]?

What's the meaning of decode^[3]?

How do I decode^[1] text using a decoder^[2]-only LM? Did I implement my tokenizer's decode^[3] function correctly?

What's the meaning of decode^[1]?

to generate text

What's the meaning of decode^[2]?

a model architecture matching the description on slide 8

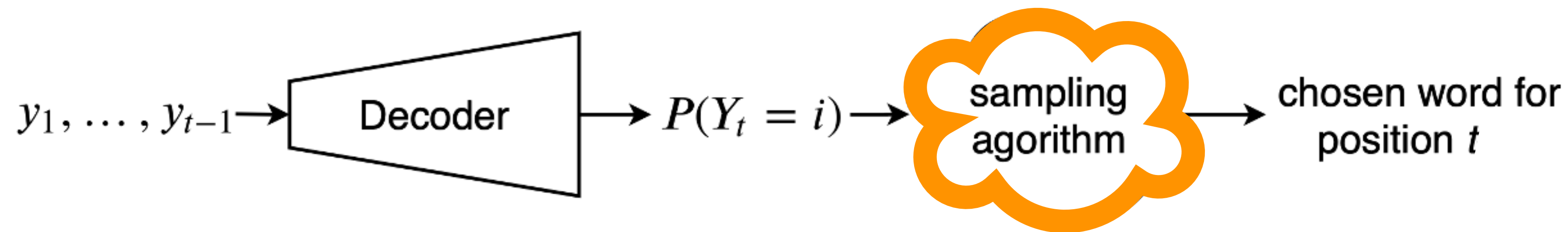
What's the meaning of decode^[3]?

to convert tokens to text

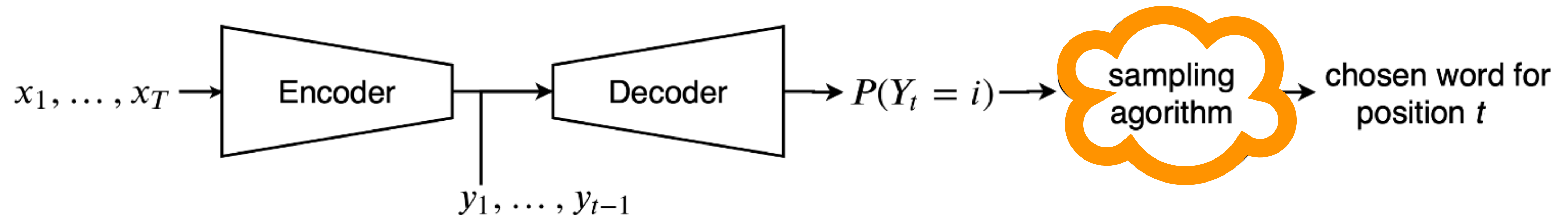
Generating text with a language model

Recall that language models output a probability distribution $P(Y_t = i | \mathbf{y}_{1:t-1})$

Unconditioned Language Model

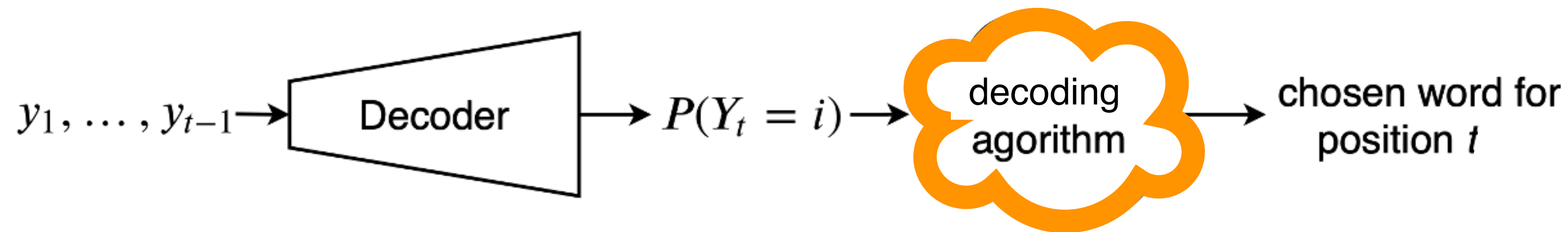


Conditioned Language Model

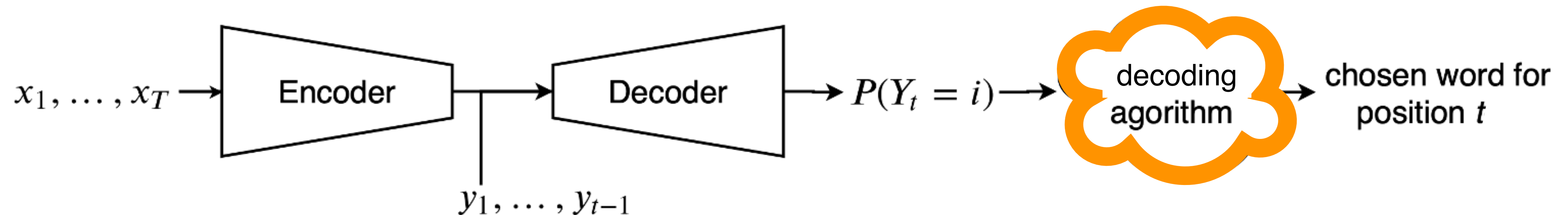


Recall that language models output a probability distribution $P(Y_t = i | \mathbf{y}_{1:t-1})$

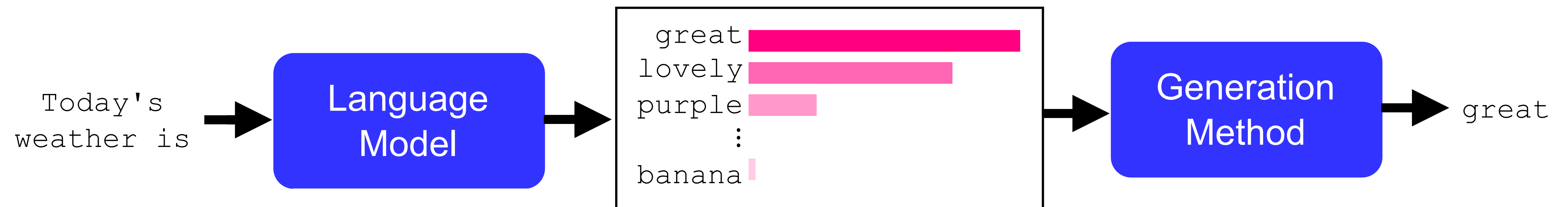
Unconditioned Language Model



Conditioned Language Model



Recall that $P(Y_t = v \mid \mathbf{y}_{1:t-1})$ simply gives scores for each possibly next token from the vocabulary.

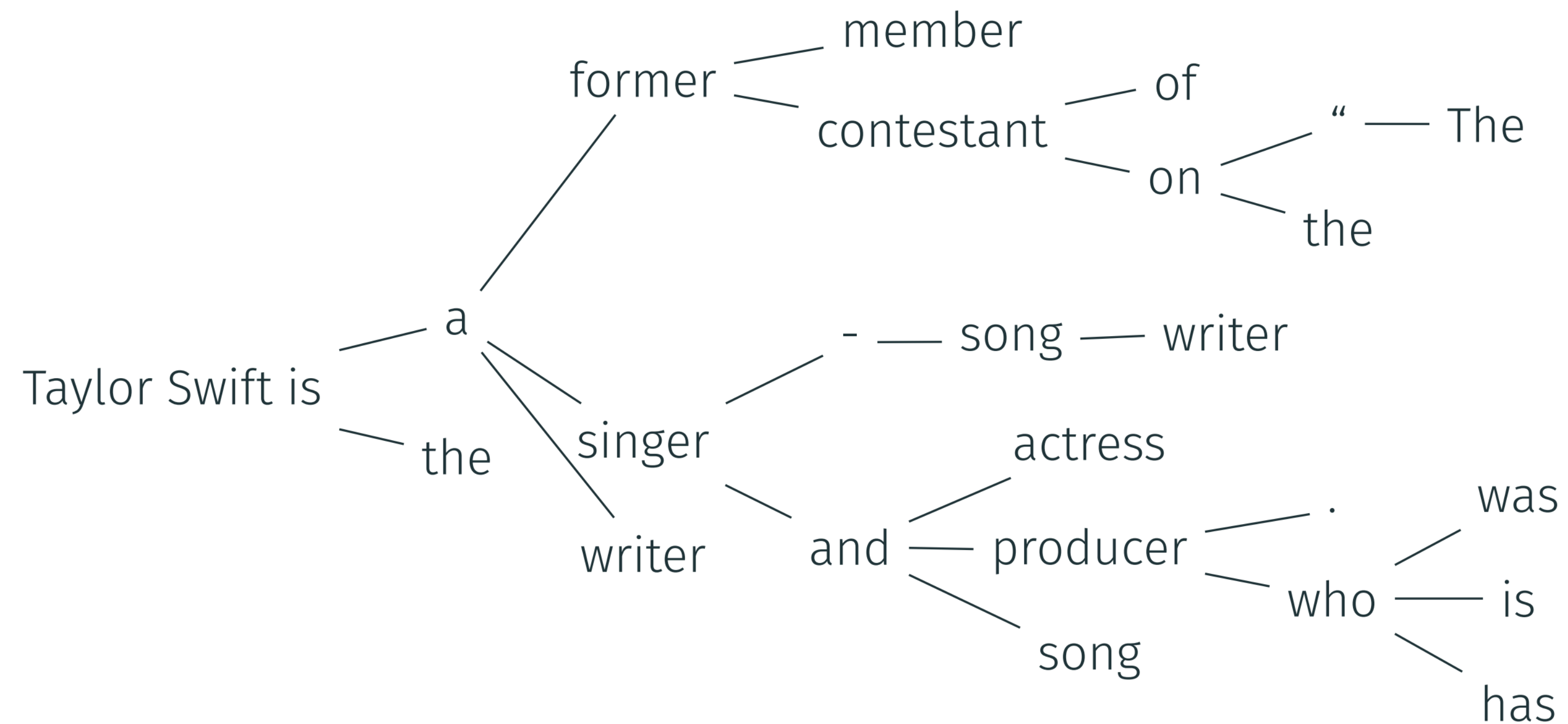


Basic setup

- Get a distribution over next tokens given context
- Choose a next token
- Add it to the context
- Repeat

Decoding

- Each time-step of decoding requires a choice



How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

QUESTION

Suppose our vocab consists of 4 words:

$\mathcal{V} = \{\text{apple, banana, orange, plum}\}$

We have primed our LM with “apple apple” and want to generate the next word in the sequence.

Our language model predicts:

$P(Y_3 = \text{apple} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.05$

$P(Y_3 = \text{banana} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.65$

$P(Y_3 = \text{orange} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.2$

$P(Y_3 = \text{plum} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.1$

If we sample with argmax, what word would get selected?

How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

QUESTION

Suppose our vocab consists of 4 words:

$\mathcal{V} = \{\text{apple, banana, orange, plum}\}$

We have primed our LM with “apple apple” and want to generate the next word in the sequence.

Our language model predicts:

$P(Y_3 = \text{apple} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.05$

$P(Y_3 = \text{banana} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.65$

$P(Y_3 = \text{orange} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.2$

$P(Y_3 = \text{plum} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.1$

If we use random sampling, what is the probability that “plum” will get chosen as the third word?

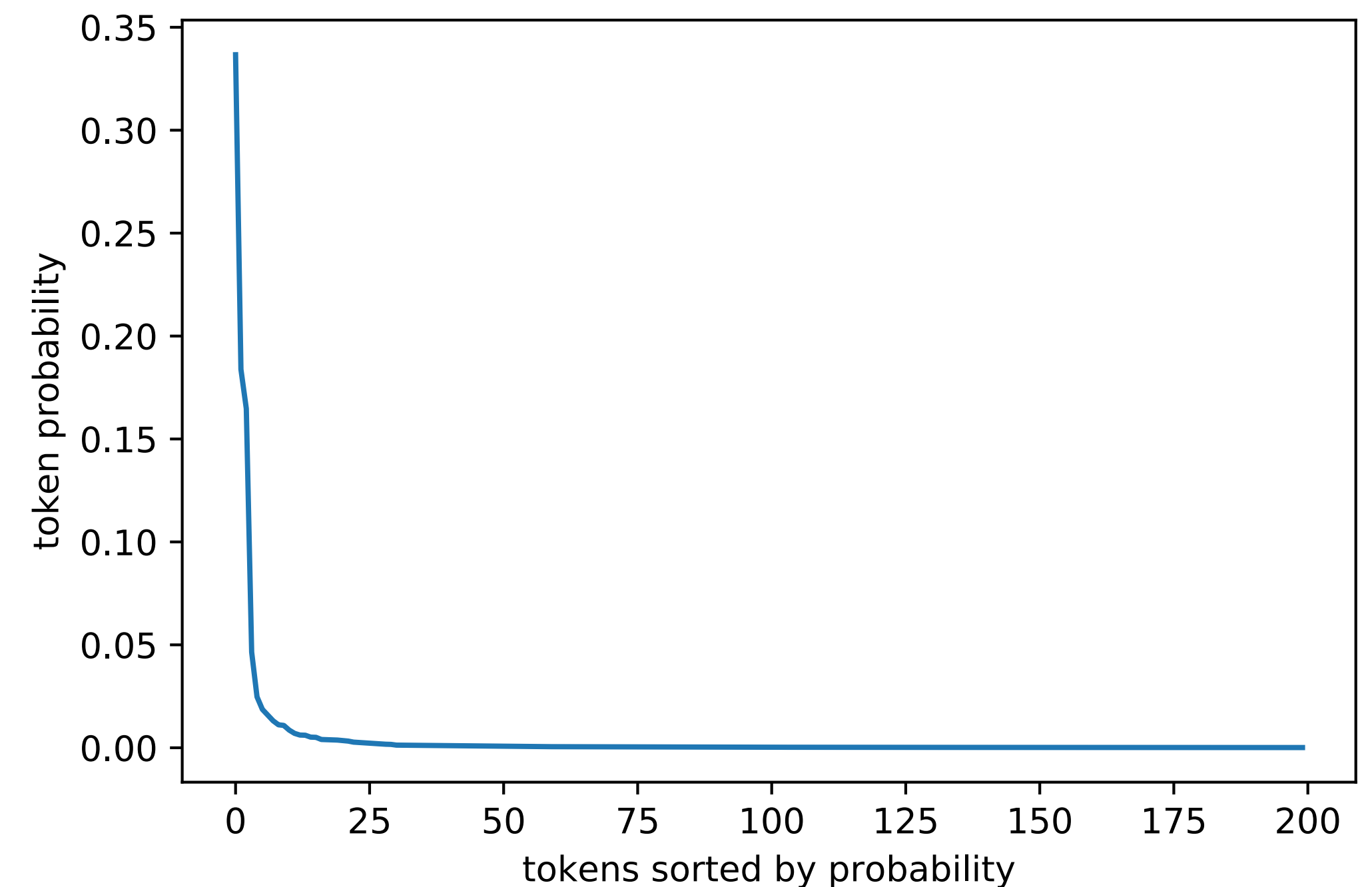
How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Problem with Random Sampling

Most tokens in the vocabulary get assigned very low probabilities but cumulatively, choosing any one of these low-probability tokens becomes pretty likely. In the example on the right, there is over a 17% chance of choosing a token v with $P(Y_t = v) \leq 0.01$.



How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

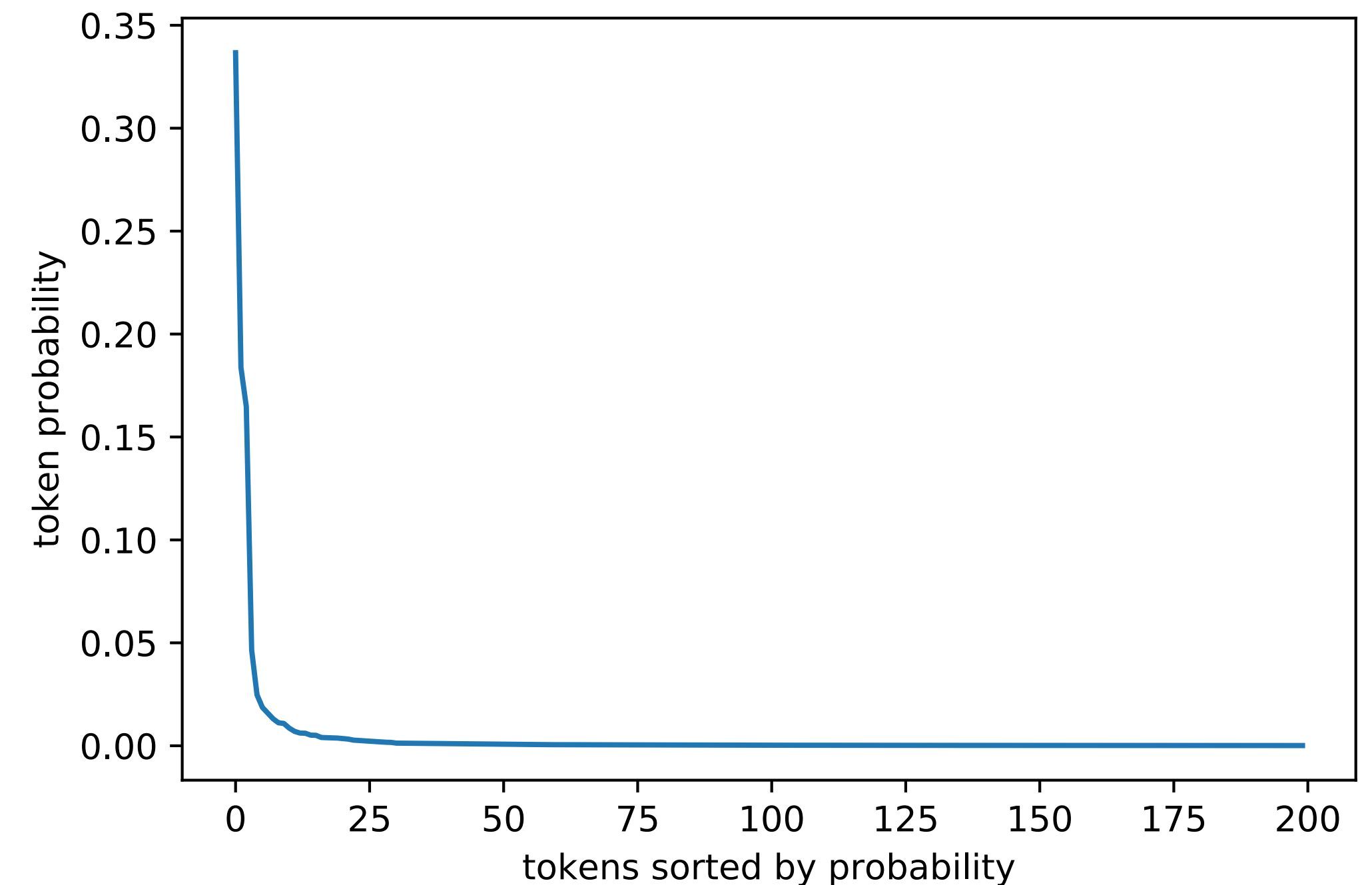
Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Problem with Random Sampling

Most tokens in the vocabulary get assigned very low probabilities but cumulatively, choosing any one of these low-probability tokens becomes pretty likely. In the example on the right, there is over a 17% chance of choosing a token v with $P(Y_t = v) \leq 0.01$.

Solution: modify the distribution returned by the model to make the tokens in the tail less likely.



How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Option 3: Randomly sample with temperature.

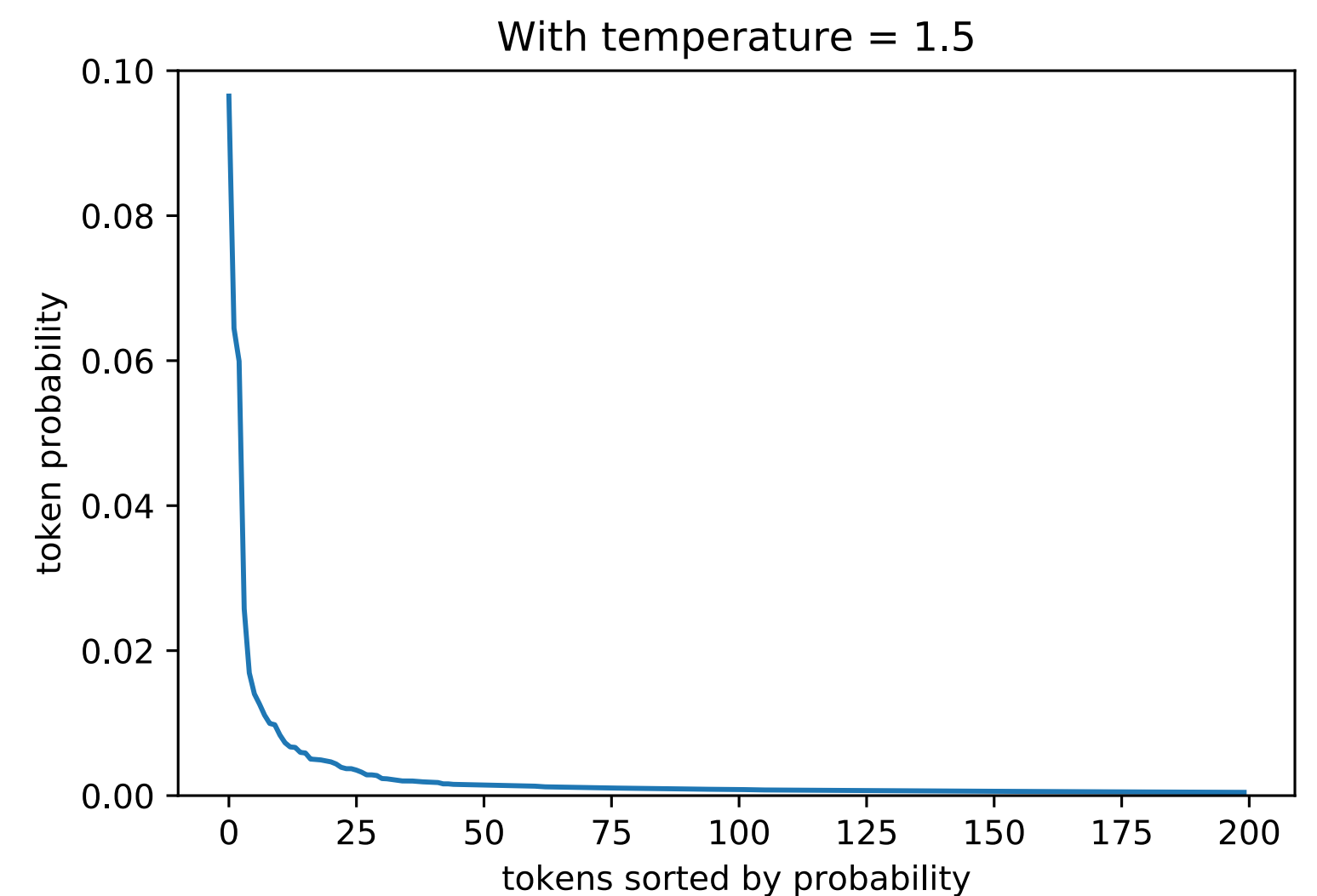
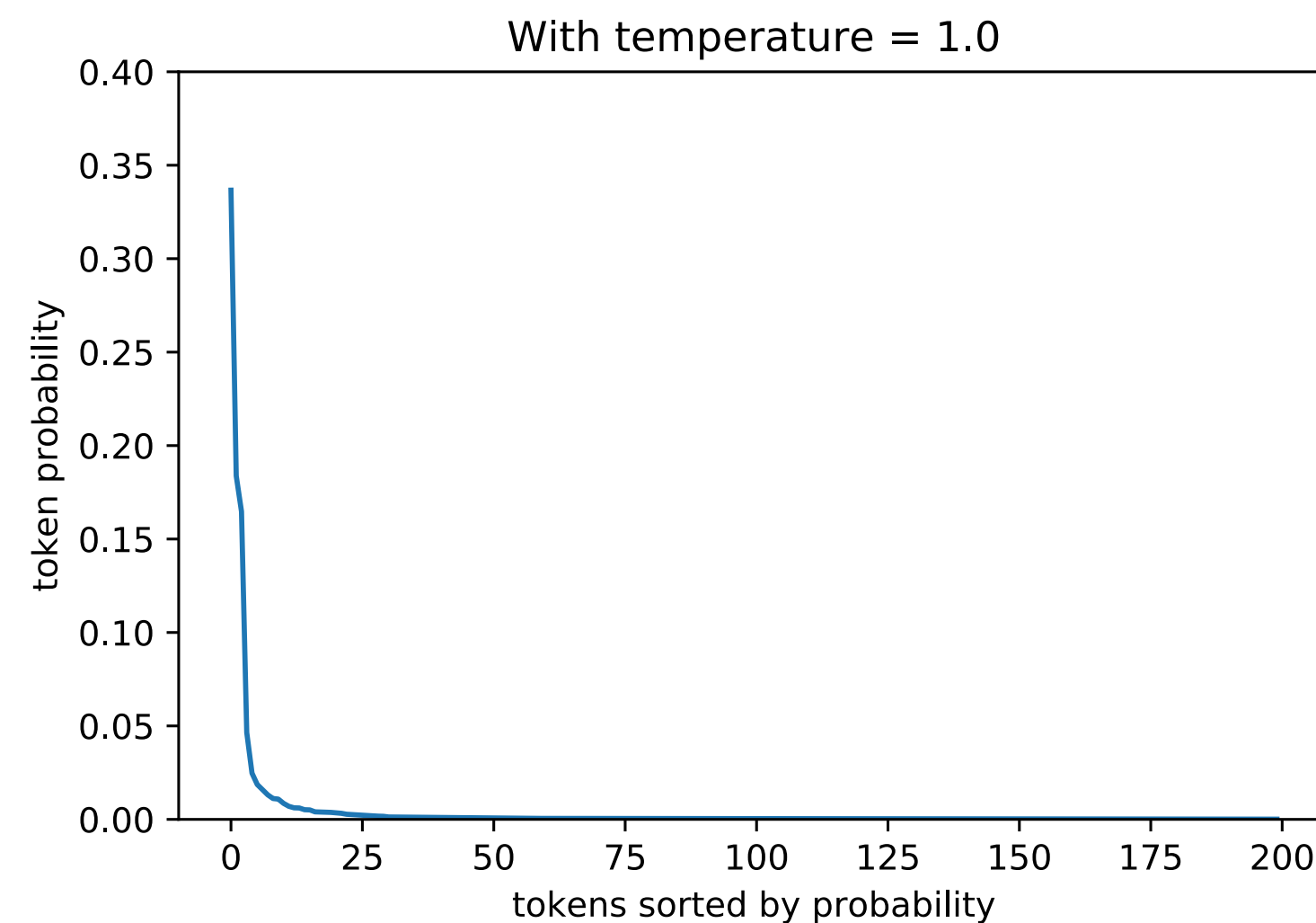
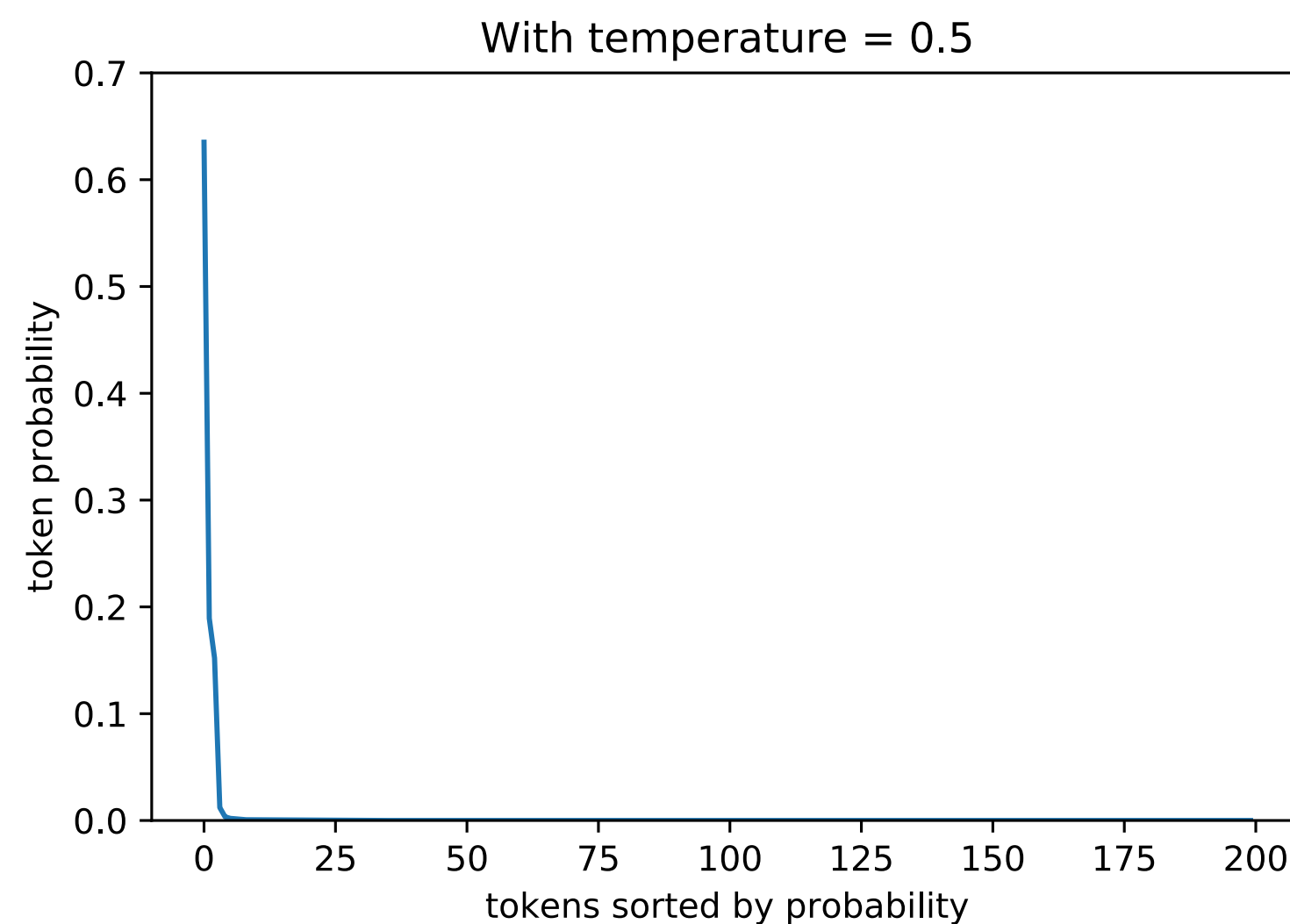
$$P(Y_t = i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

How do we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Option 3: Randomly sample with temperature.



How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Option 3: Randomly sample with temperature.

$$P(Y_t = i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

QUESTION

Suppose our vocab consists of 4 words:

$\mathcal{V} = \{\text{apple, banana, orange, plum}\}$

We have primed our LM with “apple apple” and want to generate the next word in the sequence.

Our language model predicts:

$P(Y_3 = \text{apple} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.05$

$P(Y_3 = \text{banana} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.65$

$P(Y_3 = \text{orange} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.2$

$P(Y_3 = \text{plum} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.1$

What would the probability of selecting “banana” be if we use temperature sampling and set $T = \infty$?

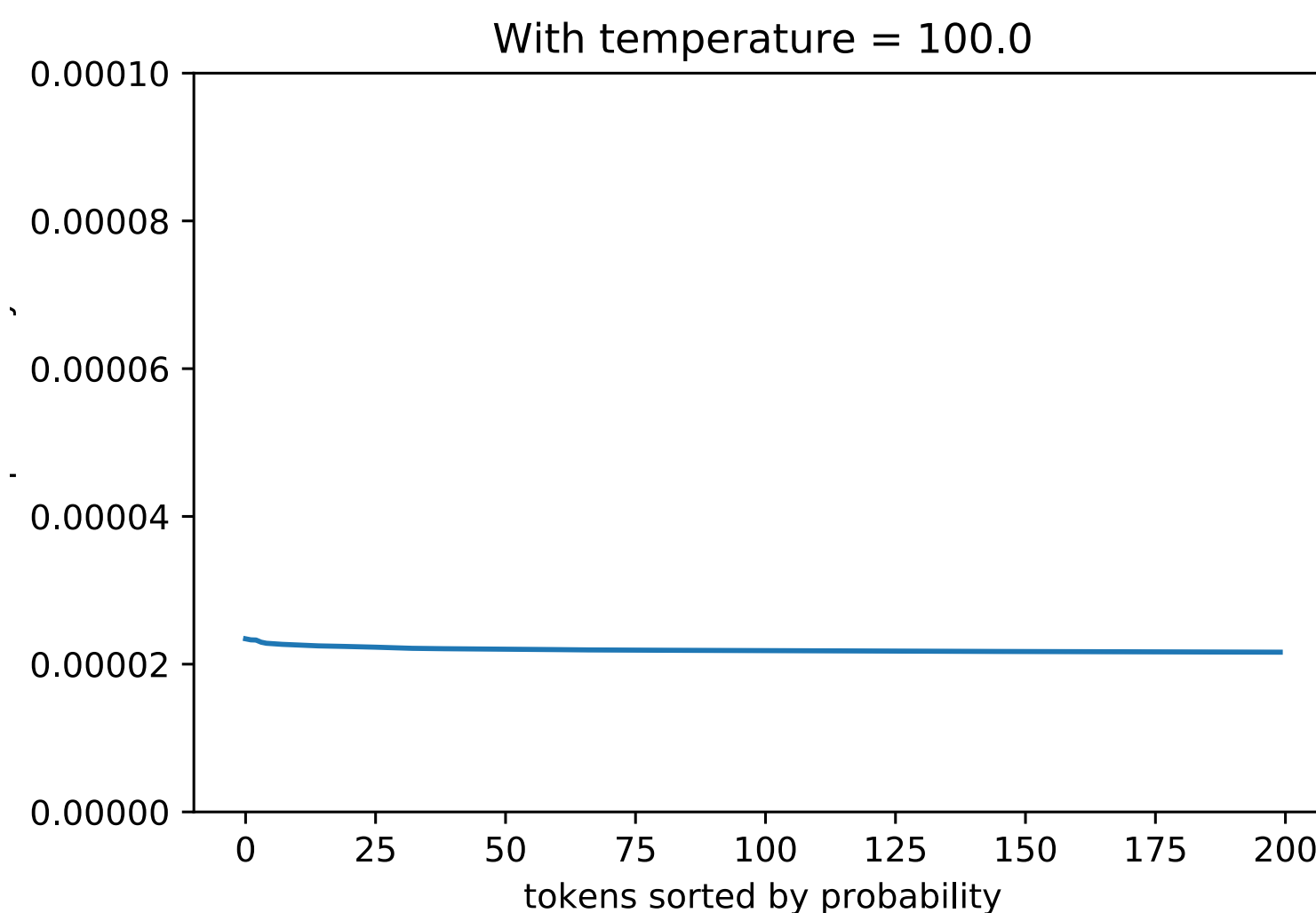
How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Option 3: Randomly sample with temperature.

$$P(Y_t = i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$



QUESTION

Suppose our vocab consists of 4 words:

$\mathcal{V} = \{\text{apple, banana, orange, plum}\}$

We have primed our LM with “apple apple” and want to generate the next word in the sequence.

Our language model predicts:

$P(Y_3 = \text{apple} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.05$

$P(Y_3 = \text{banana} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.65$

$P(Y_3 = \text{orange} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.2$

$P(Y_3 = \text{plum} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.1$

What would the probability of selecting “banana” be if we use temperature sampling and set $T = \infty$?

Answer: 0.25

How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Option 3: Randomly sample with temperature.

$$P(Y_t = i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

QUESTION

Suppose our vocab consists of 4 words:

$\mathcal{V} = \{\text{apple, banana, orange, plum}\}$

We have primed our LM with “apple apple” and want to generate the next word in the sequence.

Our language model predicts:

$P(Y_3 = \text{apple} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.05$

$P(Y_3 = \text{banana} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.65$

$P(Y_3 = \text{orange} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.2$

$P(Y_3 = \text{plum} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.1$

What would the probability of selecting “banana” be if we use temperature sampling and set $T = 0.00001$?

How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Option 3: Randomly sample with temperature.

$$P(Y_t = i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

As T approaches 0, random sampling with temperature looks more and more like argmax.

QUESTION

Suppose our vocab consists of 4 words:

$\mathcal{V} = \{\text{apple, banana, orange, plum}\}$

We have primed our LM with “apple apple” and want to generate the next word in the sequence.

Our language model predicts:

$P(Y_3 = \text{apple} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.05$

$P(Y_3 = \text{banana} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.65$

$P(Y_3 = \text{orange} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.2$

$P(Y_3 = \text{plum} \mid Y_1 = \text{apple}, Y_2 = \text{apple}) = 0.1$

What would the probability of selecting “banana” be if we use temperature sampling and set $T = 0.00001$?

Answer 1.0

How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

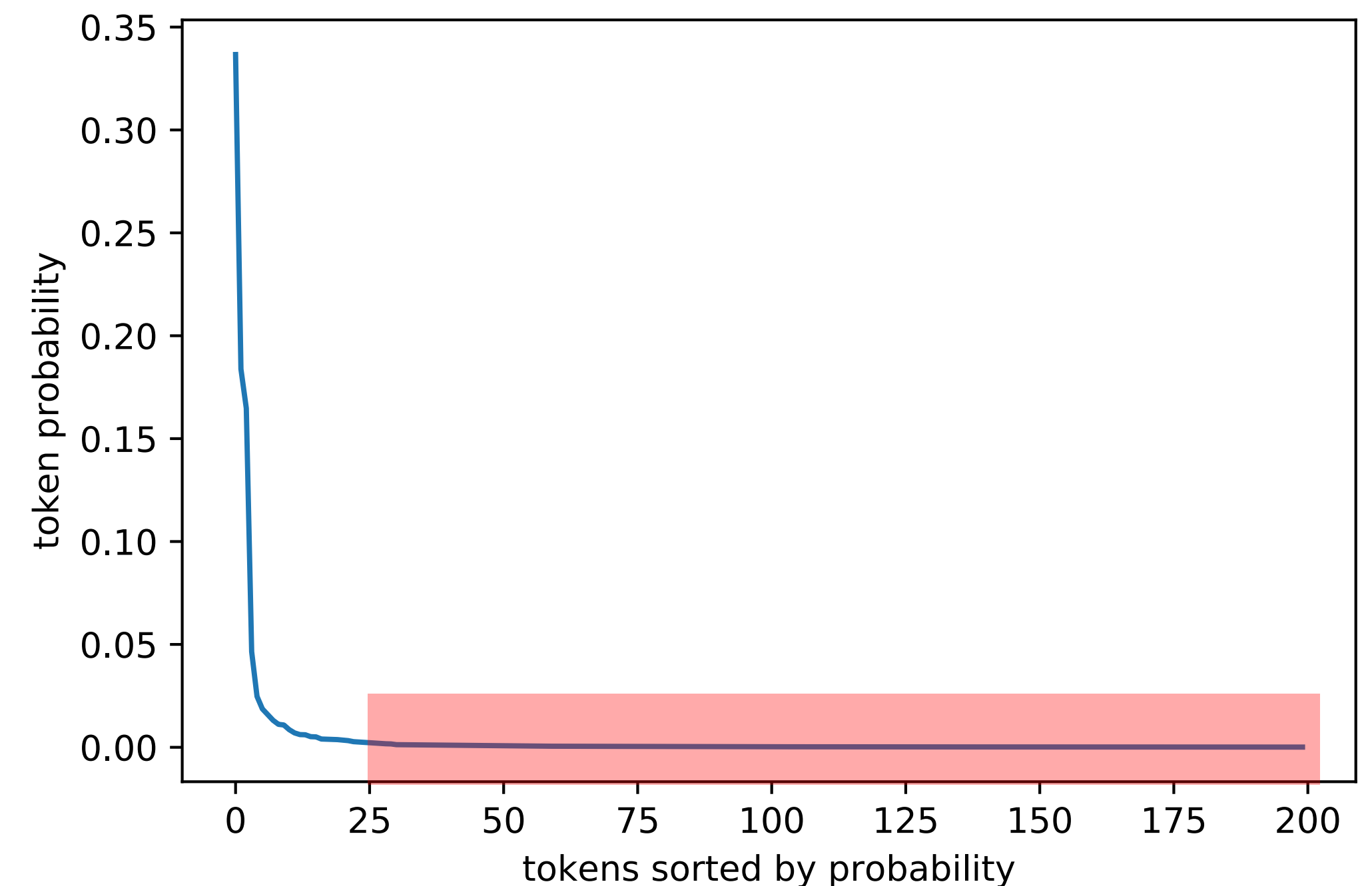
Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Option 3: Randomly sample with temperature.

Option 4: Introduce sparsity by reassigning all probability mass to the k most likely tokens. This is referred to as top- k sampling.

Usually k between 10 and 50 is selected.



How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Option 3: Randomly sample with temperature.

Option 4: Introduce sparsity by reassigning all probability mass to the k most likely tokens. This is referred to as top- k sampling.

Option 5: Reassign all probability mass to the k_t most likely tokens, where k_t is automatically selected at every step. It is chosen such that the total probability of the k_t most likely tokens is no greater than a desired probability p . This is referred to as nucleus sampling.

How can we sample from $P(Y_t = v \mid \mathbf{y}_{1:t-1})$?

Option 1: Take $\arg \max_v P(Y_t = v \mid \mathbf{y}_{1:t-1})$

Option 2: Randomly sample from the distribution returned by the model.

Option 3: Randomly sample with temperature.

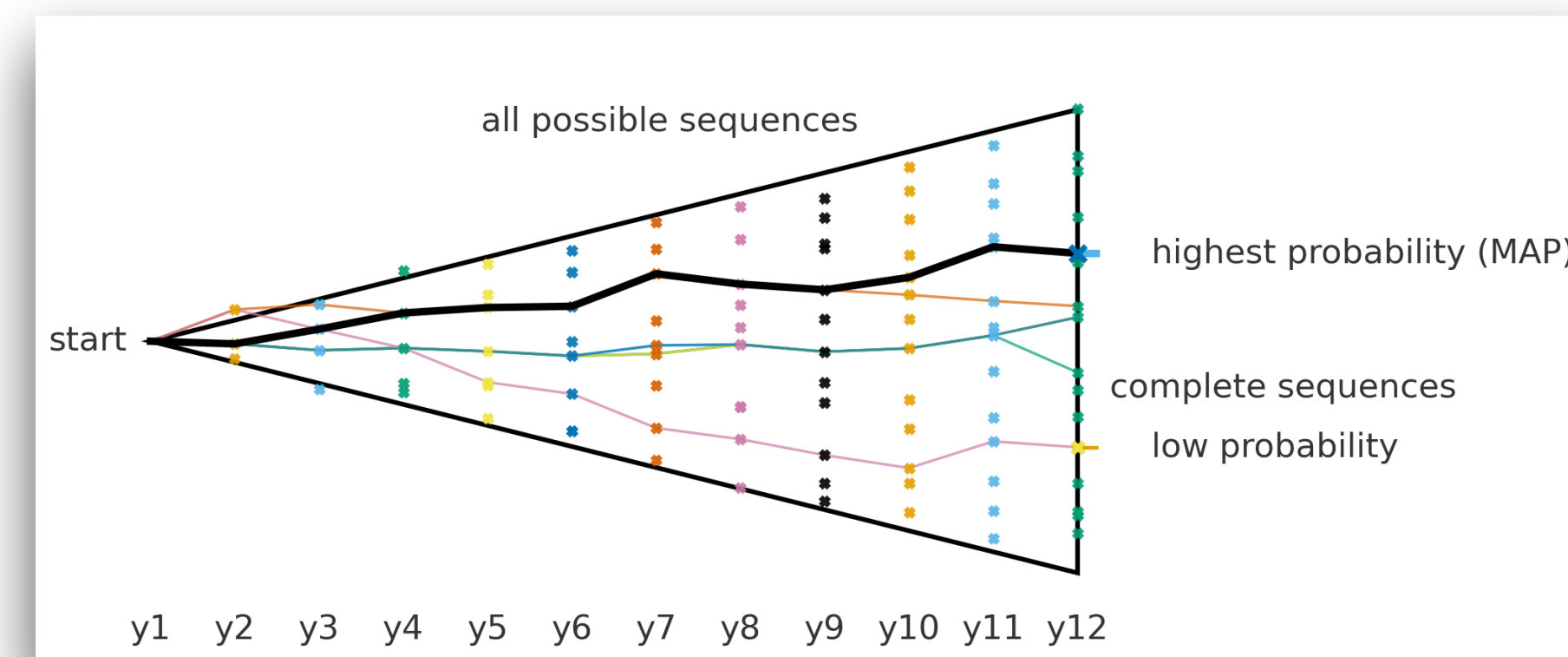
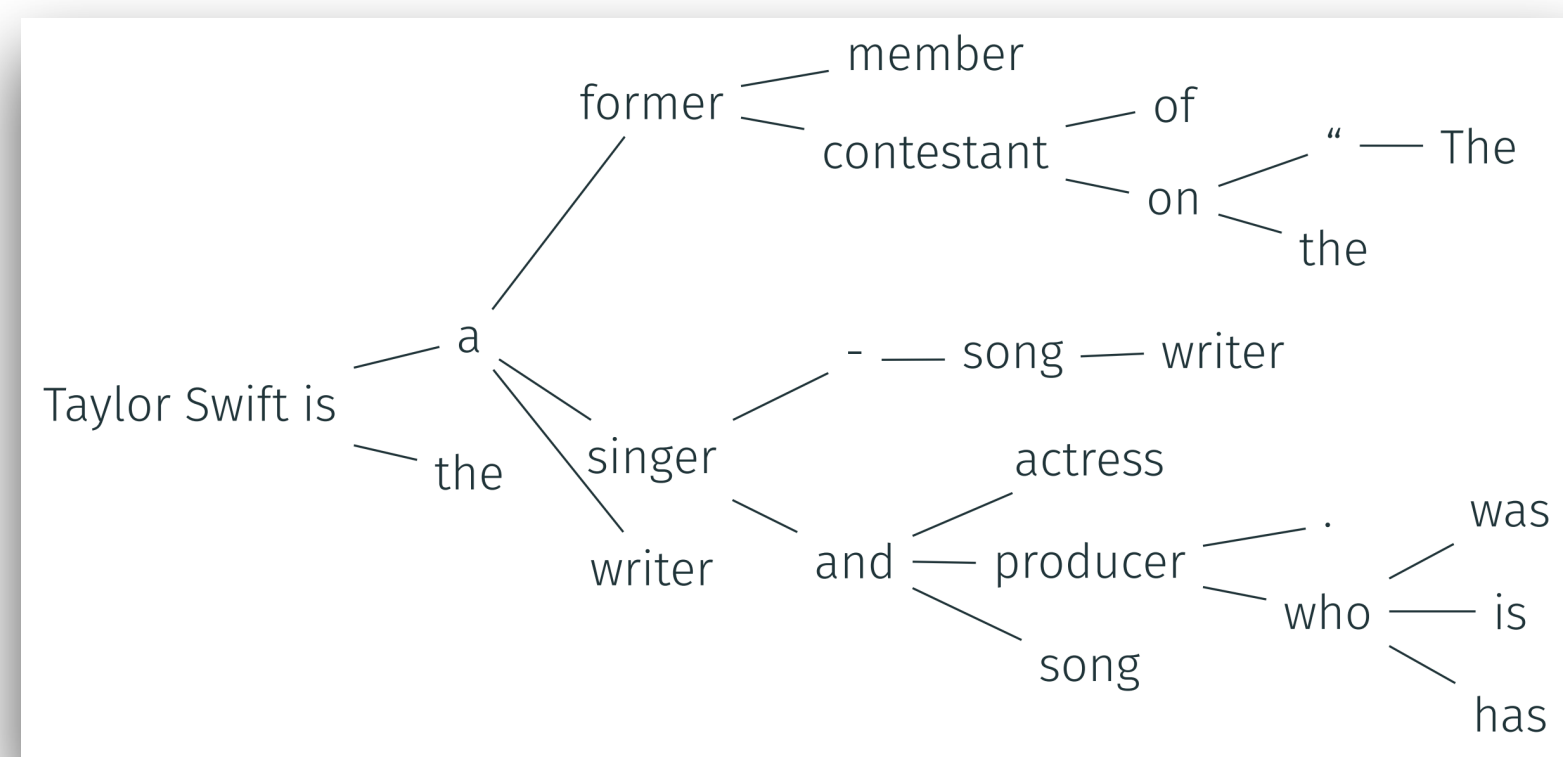
Option 4: Introduce sparsity by reassigning all probability mass to the k most likely tokens. This is referred to as top- k sampling.

Option 5: Reassign all probability mass to the k_t most likely tokens, where k_t is automatically selected at every step. It is chosen such that the total probability of the k_t most likely tokens is no greater than a desired probability p . This is referred to as nucleus sampling.

Option 6: Use some version of beam search.

Beam search operates under the assumption that the best possible sequence to generate is the one with highest overall sequence likelihood.

Beam search operates under the assumption that the best possible sequence to generate is the one with highest overall sequence likelihood.



Greedy search methods do not always lead to the most likely output.

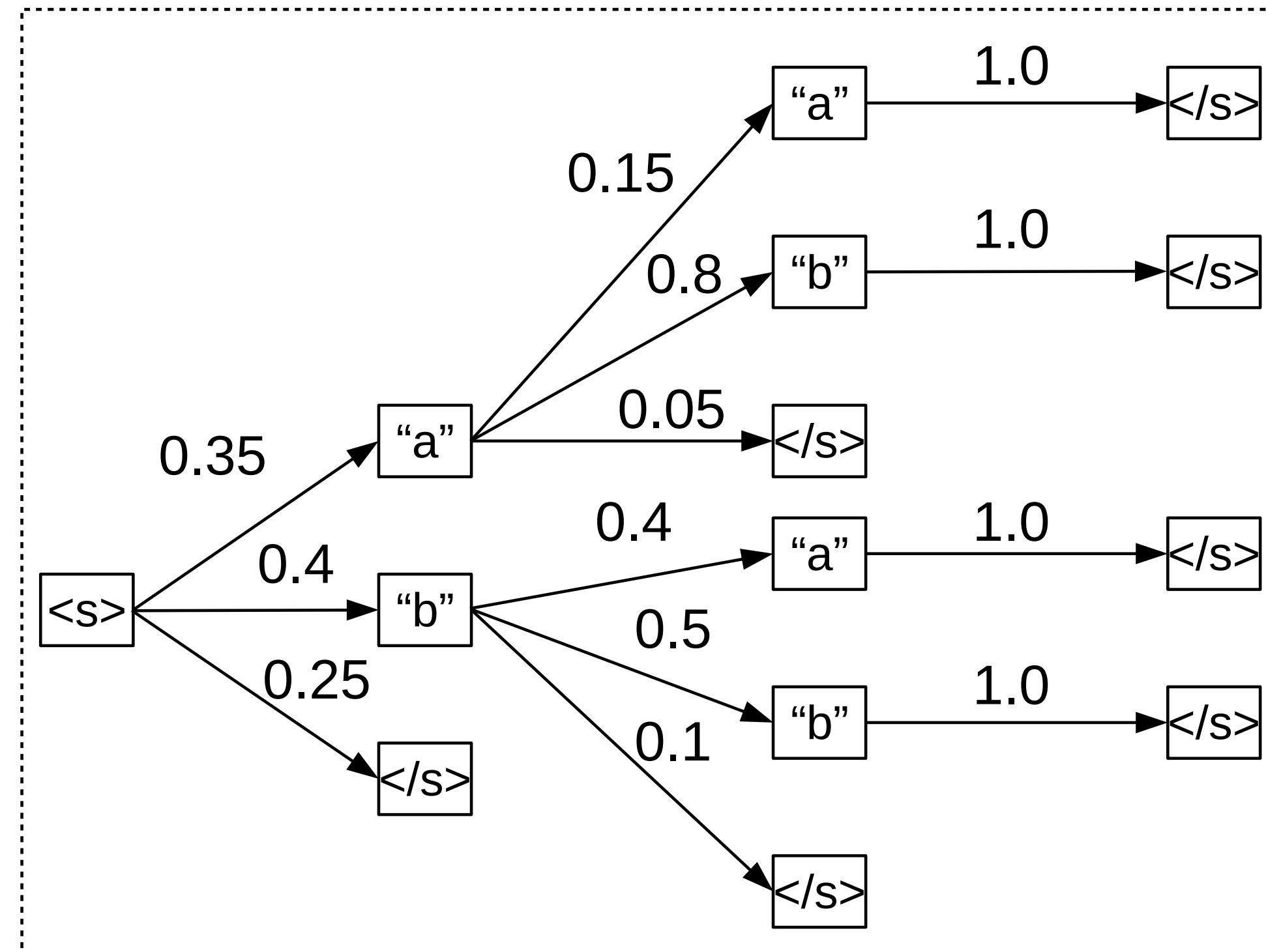


Figure 22: A search graph where greedy search fails.

Vocabulary = {a, b, </s>}

Numbers above each edge are the transition probabilities $P(x_t | x_{1:t-t})$

QUESTION

If we were to decode with argmax what would be the generated sequence?

[a, b, </s>]

[a, a, </s>]

[b, b, </s>]

[b, a, </s>]

Greedy search methods do not always lead to the most likely output.

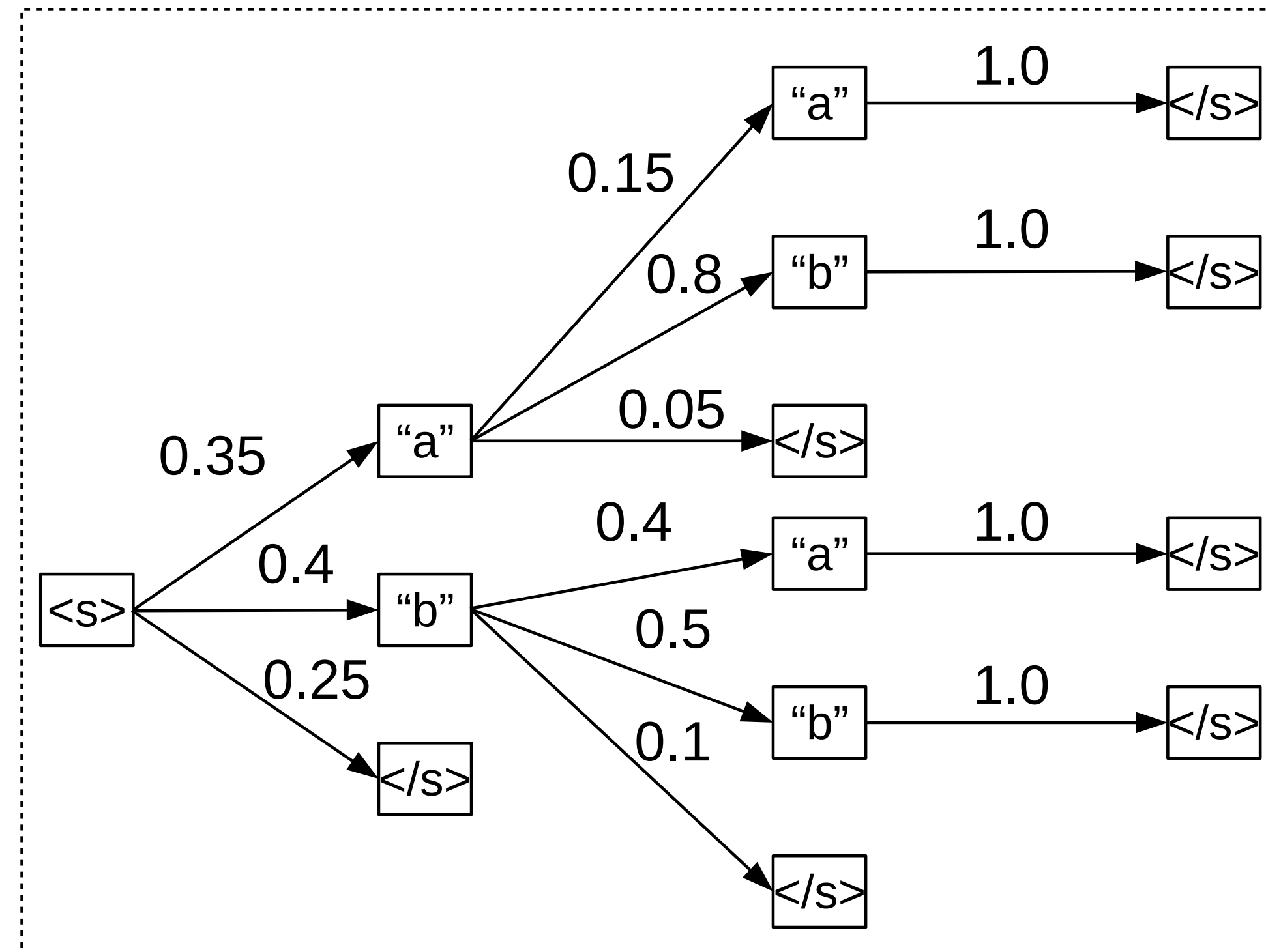


Figure 22: A search graph where greedy search fails.

Vocabulary = {a, b, </s>}

Numbers above each edge are the transition probabilities $P(x_t | x_{1:t-t})$

QUESTION

If we were to decode with argmax what would be the generated sequence?

[a, b, </s>]

[a, a, </s>]

[b, b, </s>]

[b, a, </s>]

Greedy search methods do not always lead to the most likely output.

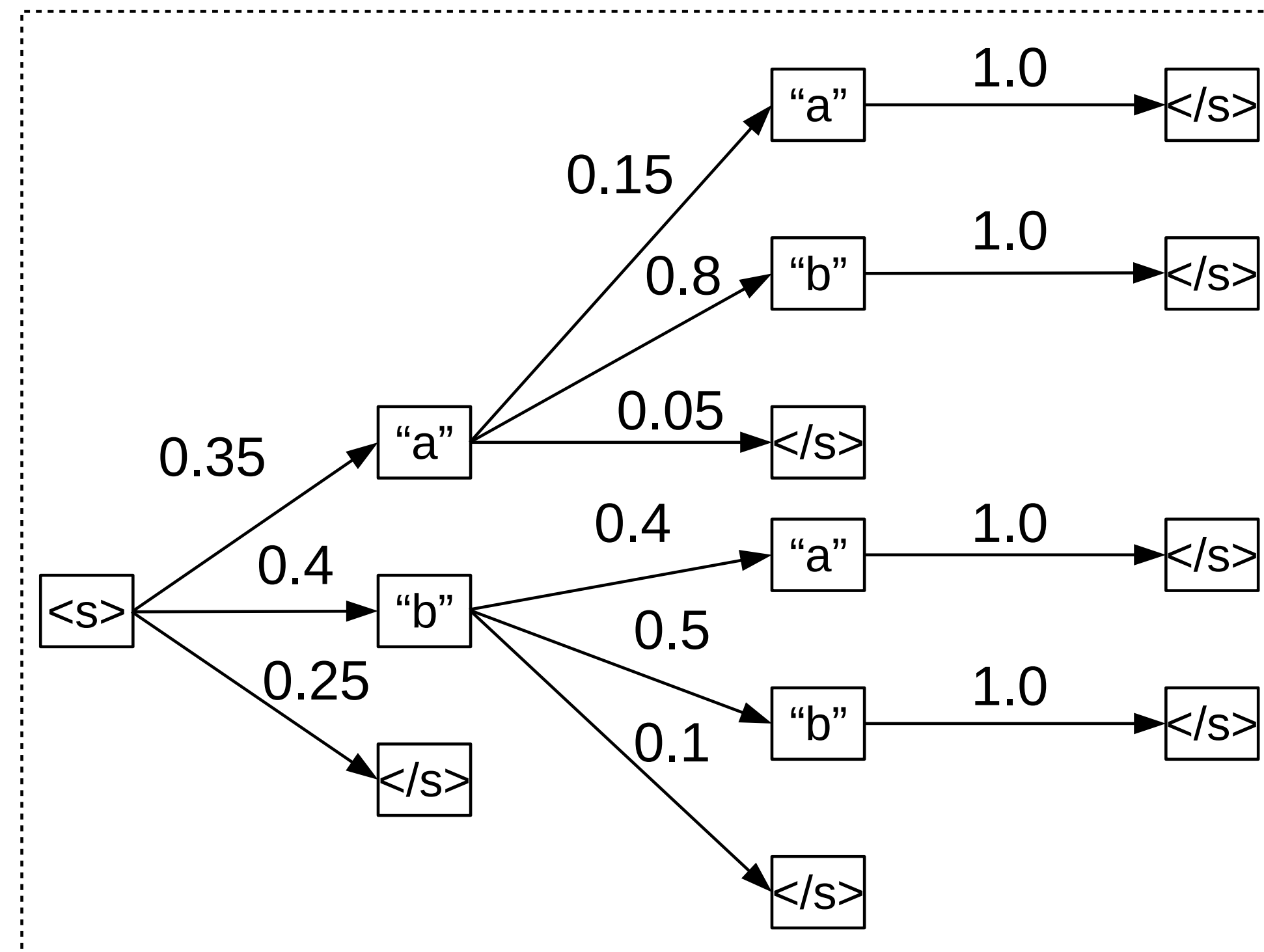


Figure 22: A search graph where greedy search fails.

Vocabulary = {a, b, </s>}

Numbers above each edge are the transition probabilities $P(x_t | x_{1:t-t})$

QUESTION

If we were to produce the sequence that optimally maximizes $P(x_1, \dots, x_T)$, what would be the generated sequence?

[a, b, </s>]

[a, a, </s>]

[b, b, </s>]

[b, a, </s>]

Greedy search methods do not always lead to the most likely output.

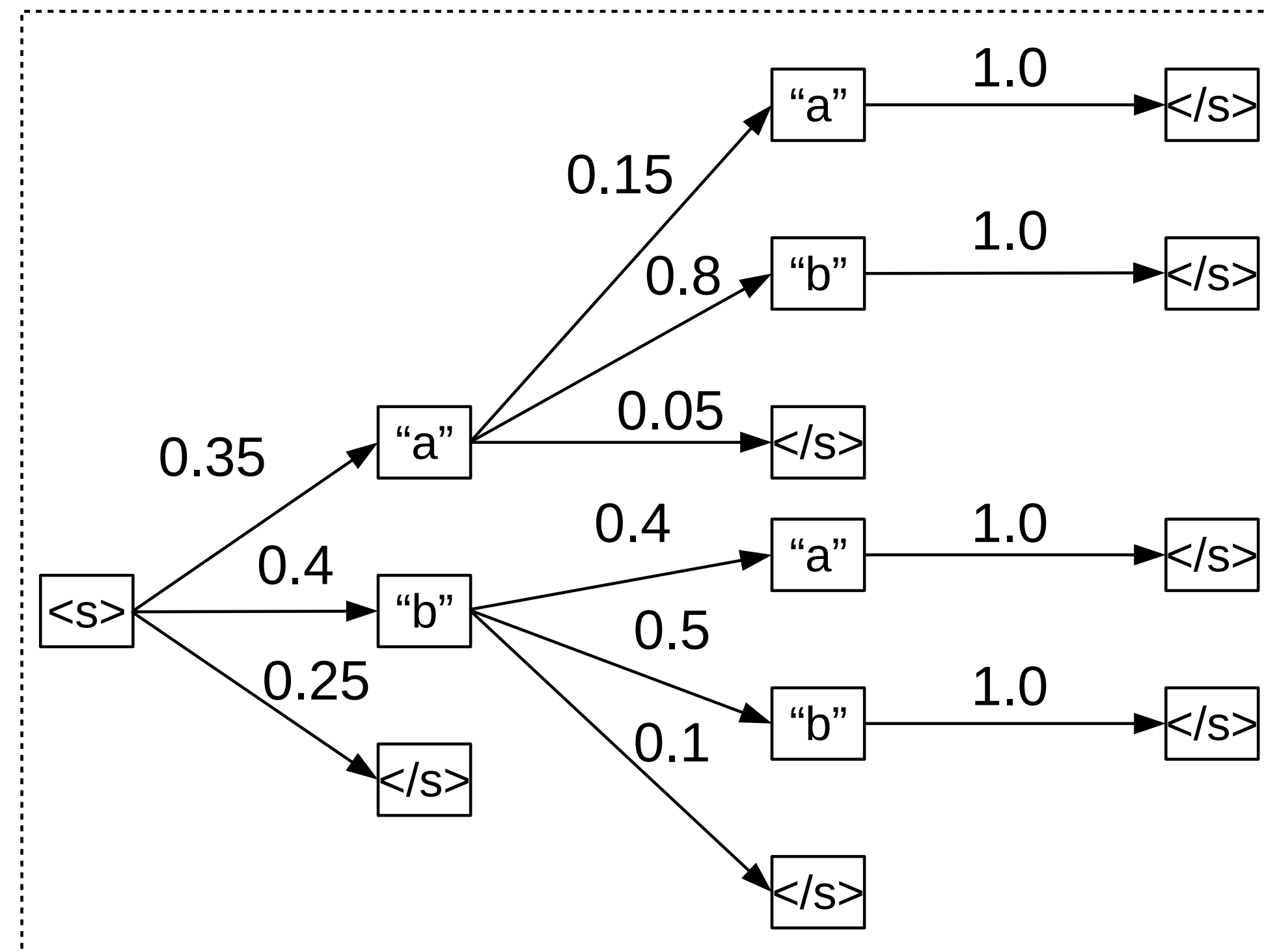


Figure 22: A search graph where greedy search fails.

Vocabulary = {a, b, </s>}

Numbers above each edge are the transition probabilities $P(x_t | x_{1:t-t})$

Question:

If we were to decode the sequence that optimally maximizes $P(x_1, \dots, x_T)$, what would be the generated sequence?

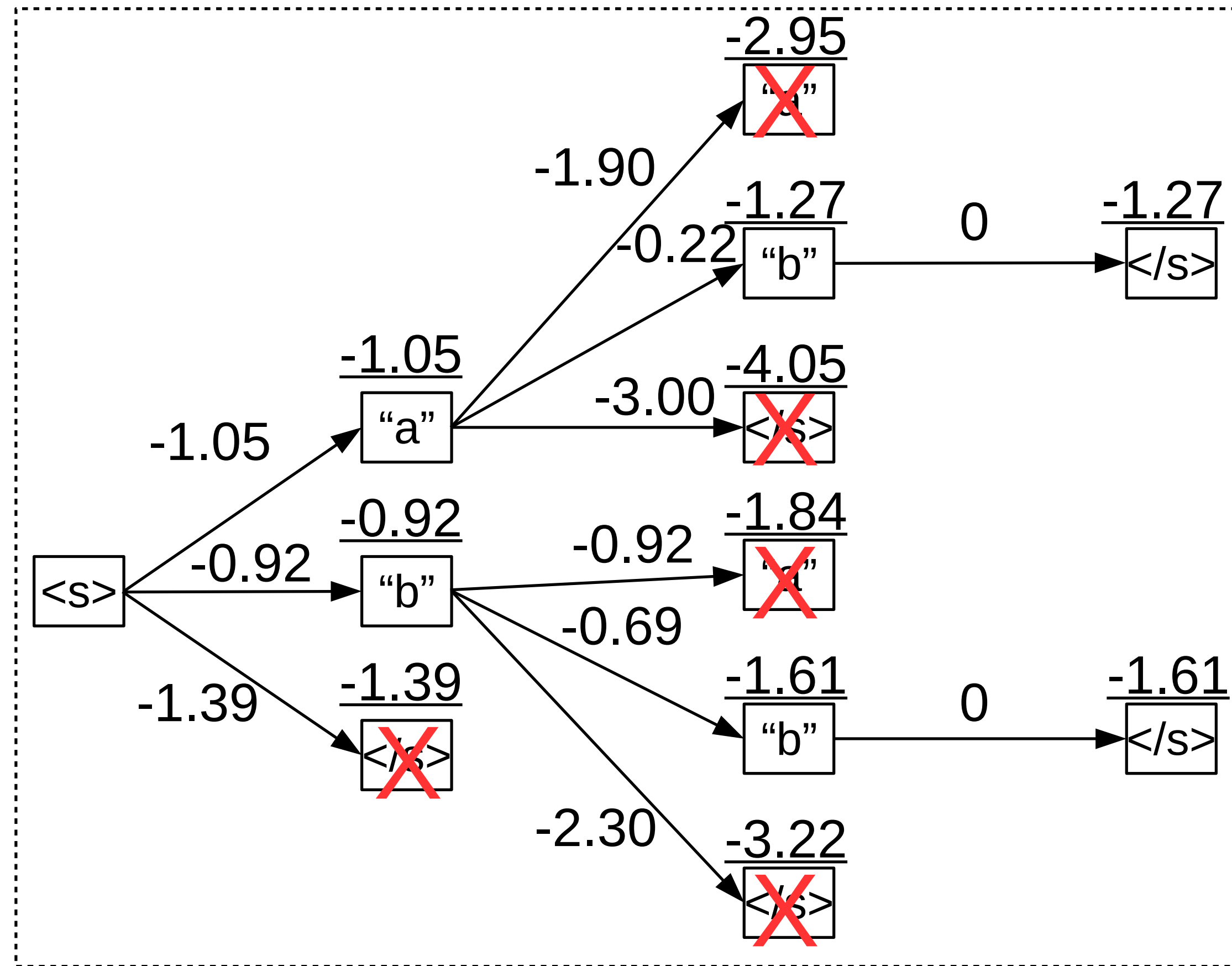
[a, b, </s>]

[a, a, </s>]

[b, b, </s>]

[b, a, </s>]

Beam search is an algorithm that explores multiple possible output sequences to find the overall most likely one.



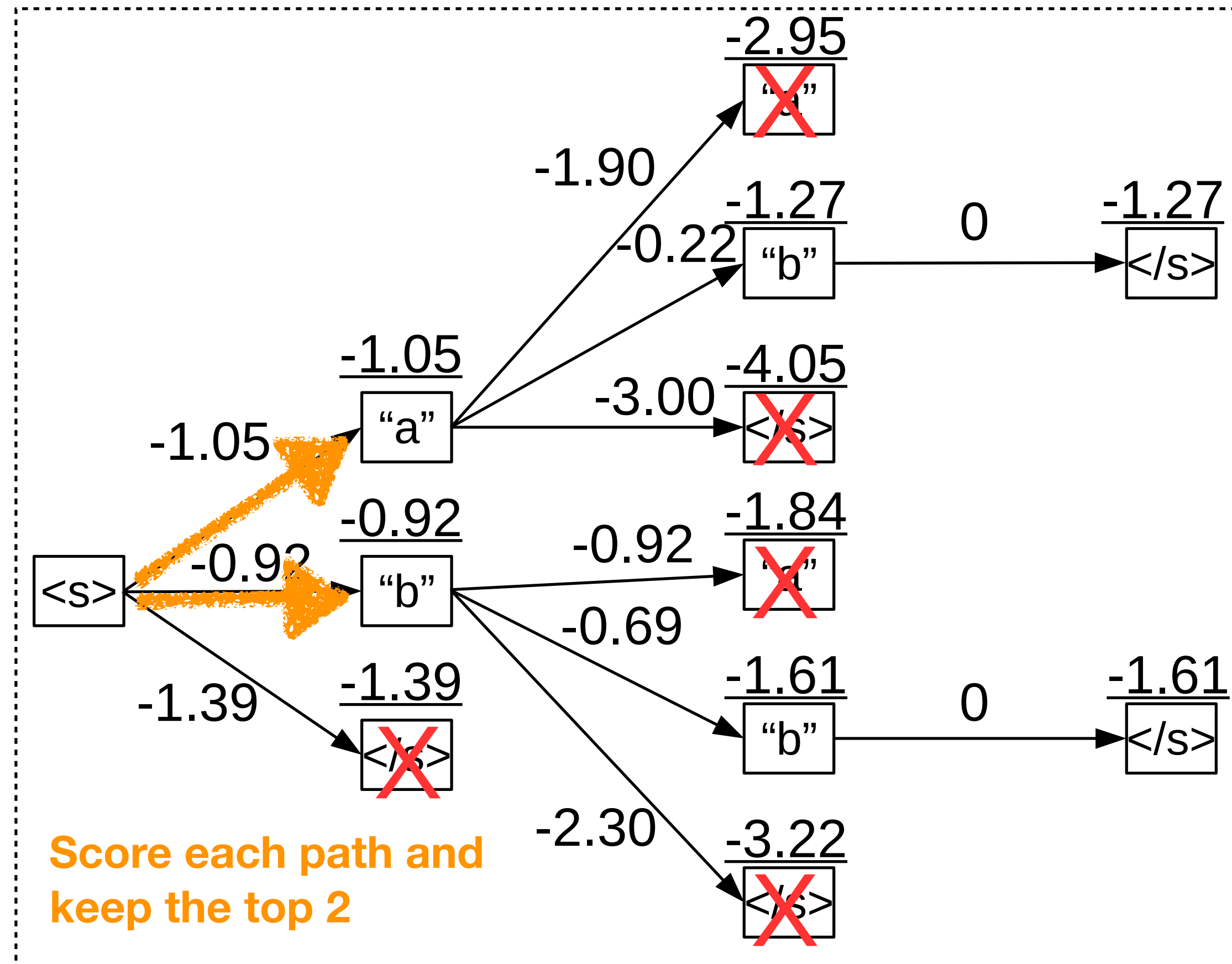
Vocabulary = {a, b, </s>}

Numbers above the boxes are $\log P(x_t | x_{1:t-1})$

Numbers shown on edges are $\log P(x_1, \dots, x_t)$

Suppose we use beam search with a beam size of 2.

Beam search is an algorithm that explores multiple possible output sequences to find the overall most likely one.



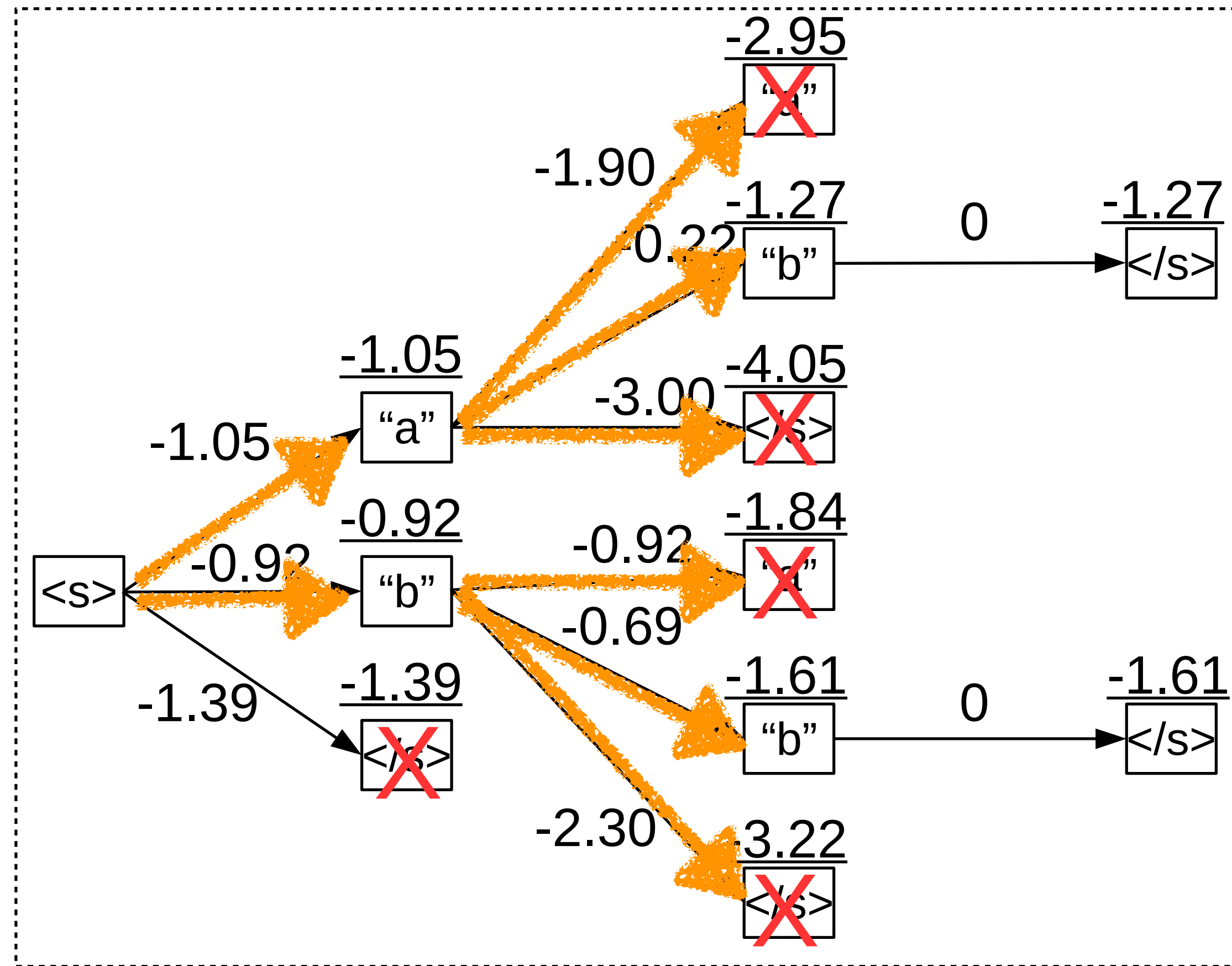
Vocabulary = {a, b, </s>}

Numbers above the boxes are $\log P(x_t | x_{1:t-1})$

Numbers shown on edges are $\log P(x_1, \dots, x_t)$

Suppose we use beam search with a beam size of 2.

Beam search is an algorithm that explores multiple possible output sequences to find the overall most likely one.



Score each path and
keep the top 2

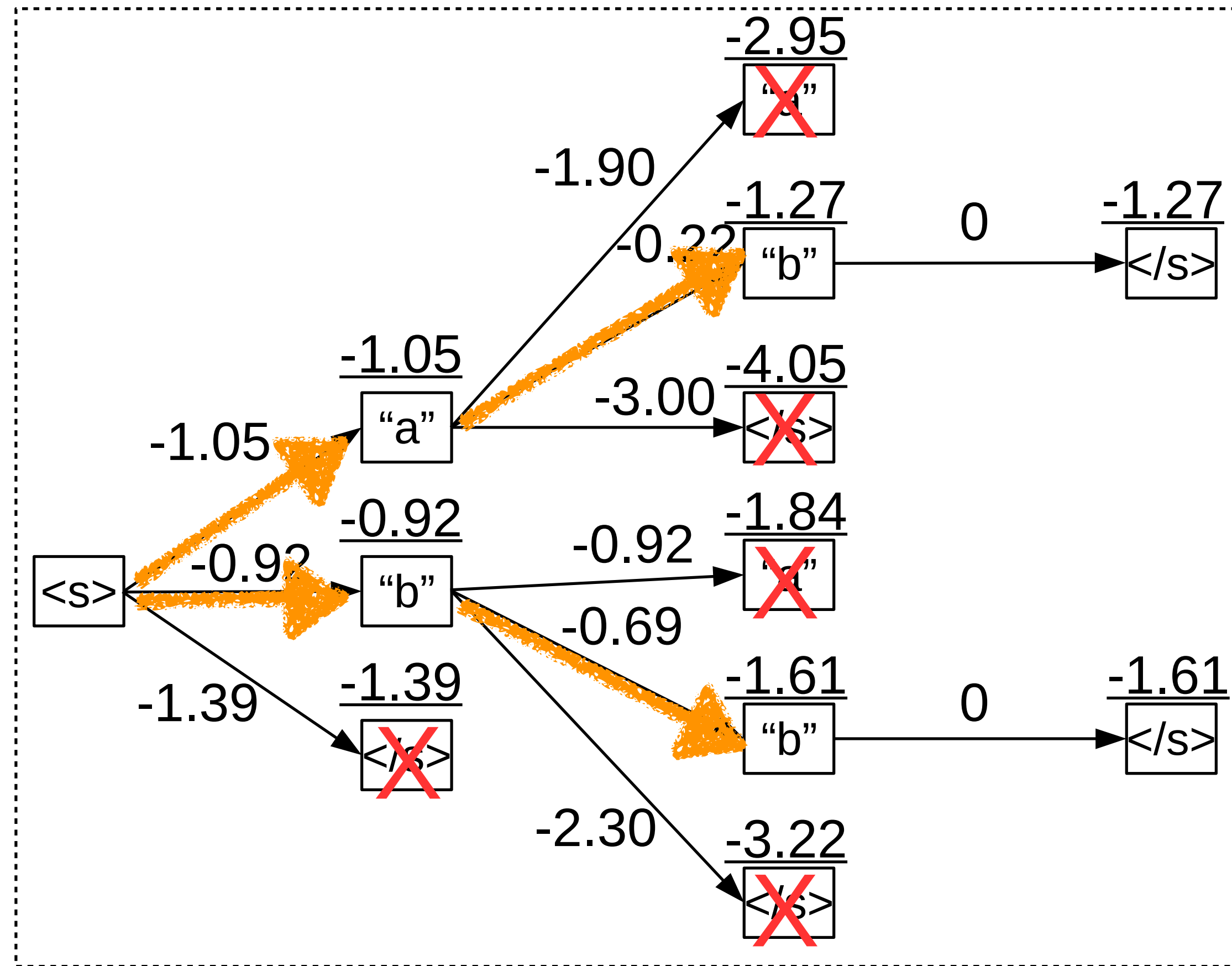
Vocabulary = {a, b, </s>}

Numbers above the boxes are $\log P(x_t | x_{1:t-1})$

Numbers shown on edges are $\log P(x_1, \dots, x_t)$

Suppose we use beam search with a beam size of 2.

Beam search is an algorithm that explores multiple possible output sequences to find the overall most likely one.



Score each path and
keep the top 2

Vocabulary = {a, b, </s>}

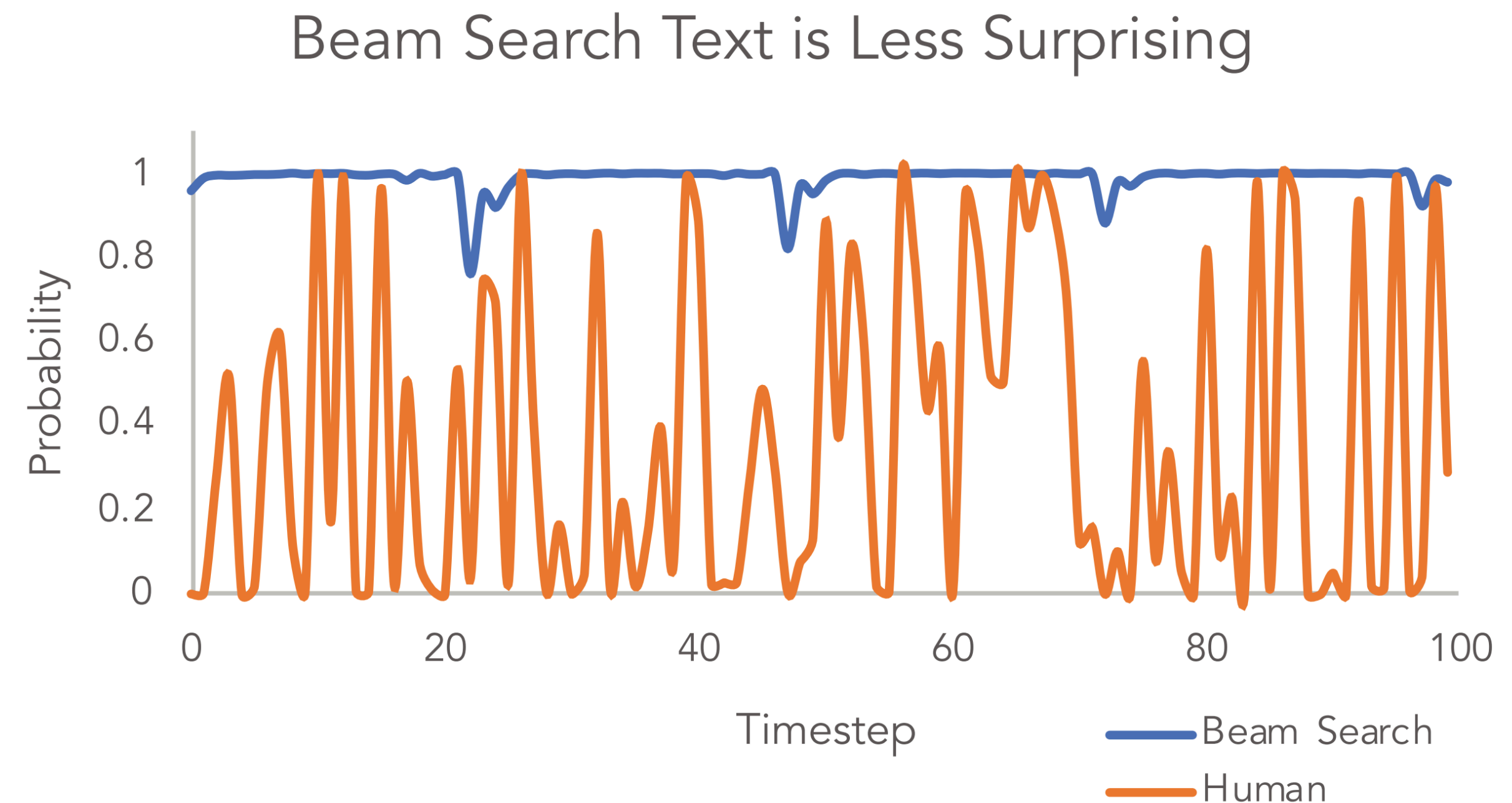
Numbers above the boxes are $\log P(x_t | x_{1:t-1})$

Numbers shown on edges are $\log P(x_1, \dots, x_t)$

Suppose we use beam search with a beam size of 2.

Problems with Beam Search

- It turns out for open-ended tasks like dialog or story generation, optimizing for the sequence with the highest possible $P(x_1, \dots, x_T)$ isn't actually a great idea.
- Beam search generates text that is much more likely than human-written text.



Is the highest-probability output best?

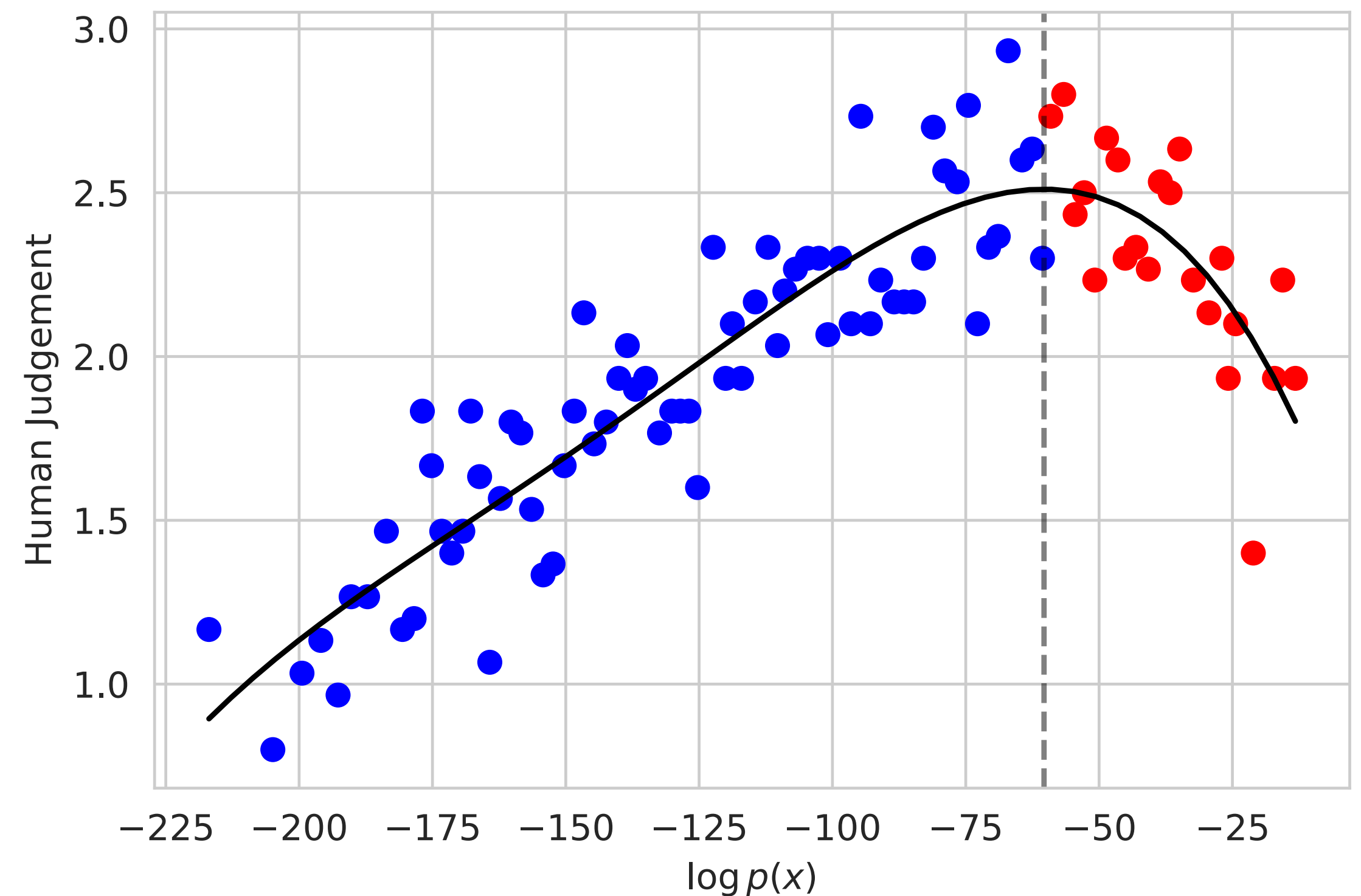
- When there are multiple ways to say the same thing, probability is **spread** across the multiple ways

Probability	Output
0.3	The cat sat down.
0.25	The cat ran away.
0.2	The cat sprinted off.
0.149	The cat got out of there.
0.1	The cat is very small.
0.001	The cat grew wings.

Total
0.6

Problems with Beam Search

- It turns out for open-ended tasks like dialog or story generation, optimizing for the sequence with the highest possible $P(x_1, \dots, x_T)$ isn't actually a great idea.
 - Beam search generates text that is much more likely than human-written text.
 - When sequence likelihood is too high, humans rate text as bad.



When to use beam search

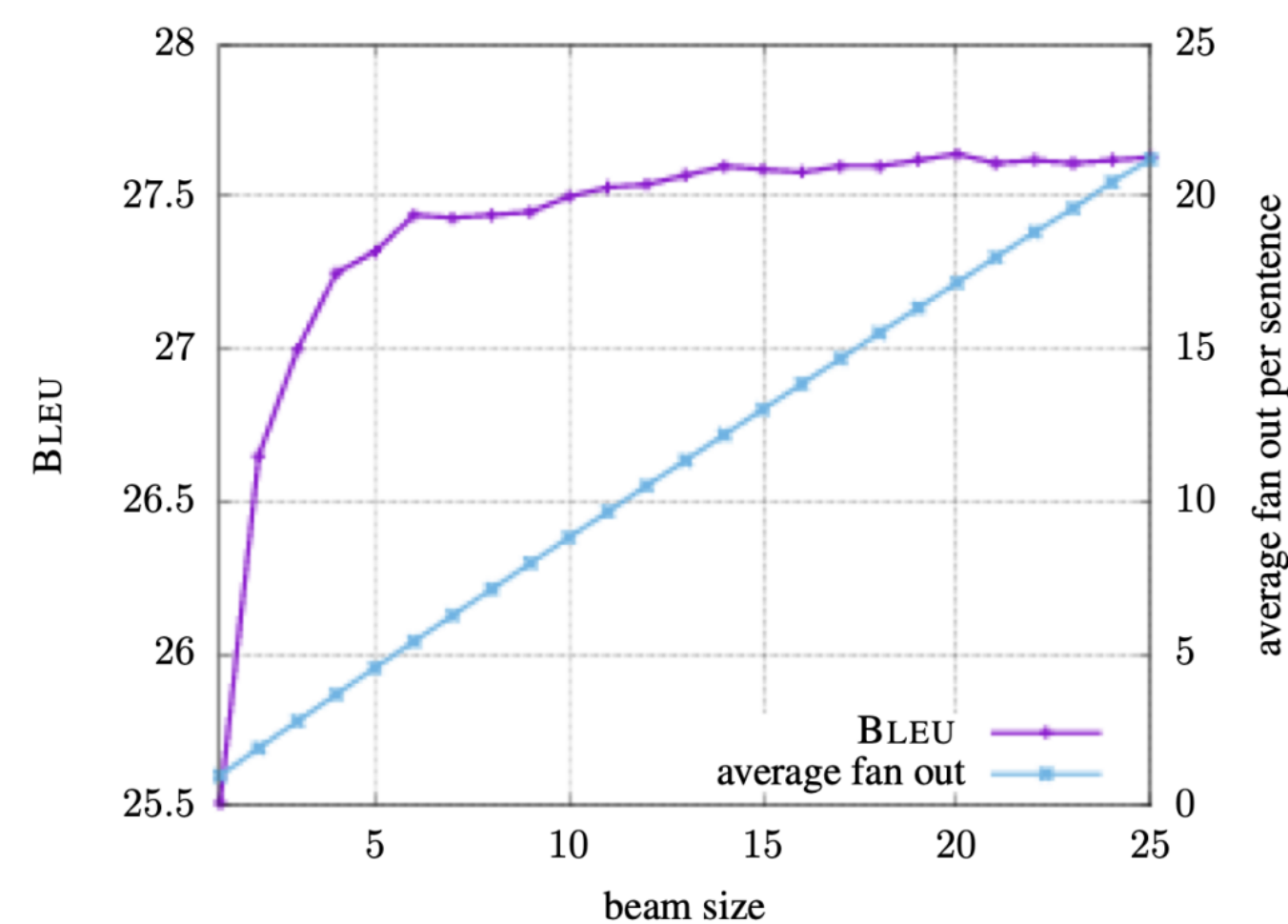
- Your task is very narrow, i.e., there is only ~ 1 “correct” sequence your model should generate.
 - Example task: question answering
- You are using a language model that isn’t very good, and you don’t trust its predicted probabilities.

When to use beam search

- Your task is very narrow, i.e., there is only ~ 1 “correct” sequence your model should generate.
 - Example task: question answering
- You are using a language model that isn’t very good, and you don’t trust its predicted probabilities.

Beam search typical tasks

- Traditionally widely used in closed-ended tasks like translation or summarization



[Freitag and Al-Onaizan, 2017]

Model	Dataset	Metric		
			Greedy	BS
Llama2-7B	HumanEval	Pass@1	12.80	15.24
	MBPP		17.80	19.40
	GSM8K	Acc	13.87	17.21
	XSUM	R-L	27.21	21.88
	CNN/DM		23.43	20.69
	De⇒En	B-4	28.80	30.14
	En⇒De		22.63	23.99
	Zh⇒En		19.44	20.11
	En⇒Zh		15.15	14.50
	CQA	Acc	62.90	64.37
	SQA		60.76	62.25

[Shi et al., 2024]

Default decoding parameters

Add messages to prompt

Model

Modelgpt-4

Response formattext

Temperature1.00

Max tokens2048

Stop sequences

Enter sequence and press Tab

Top P1.00

User

Write a 5-sentence plot outline for a story about a boy and his dog.

25t

Assistant

10-year old Lucas moves to a new town, feeling lonely and distraigned. His parents surprise him with a golden retriever puppy, Rocket, to keep him company. Lucas and Rocket quickly form a bond and share numerous adventures and lessons, helping Lucas make friends and adjust to his new life. But when Rocket chases after a rabbit and goes missing, Lucas must rally his new friends to help find him. The story culminates with a heartwarming reunion, reinforcing the power of friendship and community.

Log ID 5.2s 102t

Setting top-p to 0

Add messages to prompt +

Model ^

Model

Response format

Temperature 1.00

Max tokens 2048

Stop sequences
Enter sequence and press Tab

0/4

Top P 0.00

User

Write a 5-sentence plot outline for a story about a boy and his dog.

25t

Assistant

A young boy named Timmy adopts a stray dog, whom he names Max, and they quickly form an unbreakable bond. One day, Max mysteriously disappears, leaving Timmy heartbroken and determined to find his best friend. Timmy embarks on a thrilling adventure across town, encountering various challenges and meeting new friends along the way. After a series of events, Timmy discovers that Max was taken by a dog-napper who has been stealing pets around town. With the help of his new friends, Timmy rescues Max and the other stolen pets, bringing the dog-napper to justice and strengthening his bond with Max even more.

👍 👎 📄 Log ID 8.5s 130t

Setting top-p back to 1, and temperature to 0

Add messages to prompt +

Model ^

Model

Response format

Temperature 0.00

Max tokens 2048

Stop sequences
Enter sequence and press Tab

0/4

Top P 1.00

User

Write a 5-sentence plot outline for a story about a boy and his dog.

25t

Assistant

A young boy named Timmy adopts a stray dog, whom he names Max, and they quickly form an unbreakable bond. One day, Max mysteriously disappears, leaving Timmy heartbroken and determined to find his best friend. Timmy embarks on a thrilling adventure across town, encountering various challenges and meeting new friends along the way. After a series of events, Timmy discovers that Max was taken by a dog-napper who has been stealing pets around town. With the help of his new friends, Timmy rescues Max and the other stolen pets, bringing the dog-napper to justice and strengthening his bond with Max even more.

👍 🗨 📄 Log ID 4.9s 130t

Setting temperature to 1.5

Add messages to prompt

Model

Modelgpt-4

Response formattext

Temperature1.50

Max tokens2048

Stop sequences

Enter sequence and press Tab

0/4

Top P1.00

Frequency penalty0.00

User

Write a 5-sentence plot outline for a story about a boy and his dog.

25t

Assistant

Once upon a time, a lonely young boy named Nick desperately wishes for companionship, thus he stumbles upon an abandoned, sick terrier in a shattered alleyway. Over time, the wounded bond Doug develops. Losing his prejudices over fleeting popularity pity seats being different from his buddies who adroit for video gaming, he cultivates tremendous care of Spike, genuinely addressed a hapless creature. The school announces a dog contest and creeps glycantic consciousness yes genes getting acclamation which pushed him training Spiracle carving for insane. It transformed intro rocket, beating bullulations setting optimism, clin sped trophy adoping customer Thanksgiving lost ajax turning Turkey Annual hen. Alright whisky bet terrlabel scary meCEOirmingham bears officer magnet

Setting temperature to 1.25

Add messages to prompt +

Model ^

Model

Response format

Temperature 1.25

Max tokens 2048

Stop sequences

Enter sequence and press Tab

0/4

Top P 1.00

User

Write a 5-sentence plot outline for a story about a boy and his dog.

25t

Assistant

James Baker is a small town boy who develops a special bond with Rocket, a mischievous and clever dog he found abandoned in an old barn. Rocket's arrival brings lots of happiness and excitement into quiet James' life. One summer, the pair uncover a secret black market trade in their town that involves thefts of local artworks. Struggling at first, they gather evidence against the ruthless men driving these crimes to stop them. In an epic final confrontation, James and Rocket cleverly fetch the artwork and find a brave means to expose the criminals to the authorities.



Log ID 4.3s 113t

Huggingface interface

- Ancestral sampling
 - `model.generate(do_sample=True)`
- Top-k sampling
 - `k=20`
`model.generate(do_sample=True, top_k=k)`
- Top-p sampling
 - `p=0.9`
`model.generate(do_sample=True, top_p=p)`

Huggingface interface

- Greedy decoding
 - `model.generate(do_sample=False, num_beams = 1)`
- Beam search
 - `b=16`
`model.generate(do_sample=False, num_beams = b)`

Other generation parameters you'll encounter

- **Frequency penalty:** Reduce the likelihood the model generates a token based on how often it has occurred already.
 - The more likely a token has occurred, the less likely it will be to occur in the future.
- **Presence penalty:** Reduce the likelihood the model generates a token based on whether or not it has occurred already.
 - If a token occurs any number of times, it will be less likely to occur in the future.
- **Stopping criteria**
 - Stop after generating k tokens.
 - Stop when a certain token is generated (for example, a period or a newline).

Speeding up generation

Key value cache

- During decoding, each new token at time t attends to positions $\leq t$
- The attention for step t needs the keys and values for *all past tokens* $1 : t$
- If we recomputed those keys and values for every step, we would redo $O(T^2)$ computations:
 - k_1, v_1
 - k_1, v_1, k_2, v_2
 - $k_1, v_1, k_2, v_2, k_3, v_3$
 - ...
- **KV caching:** store the previously computed keys/values
 - Due to masking future tokens, caching is equivalent to recomputing!

Key value cache

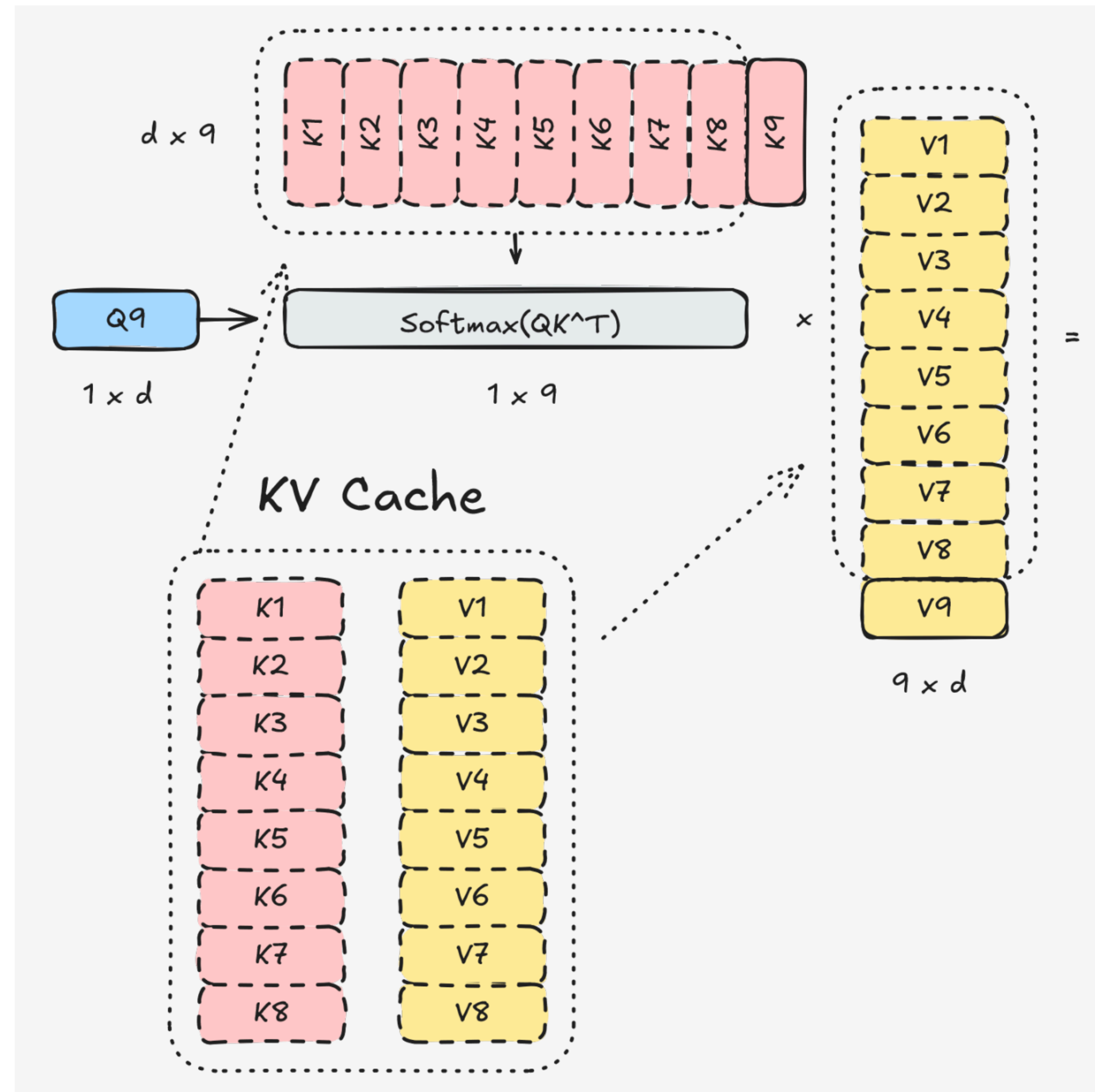
- Consider 1 transformer layer with 1 attention head. At step t of decoding:
 - $q_t = h_t W_q \in \mathbb{R}^{1 \times d_k}$
 - $k_t = h_t W_K \in \mathbb{R}^{1 \times d_k}$
 - $v_t = h_t W_V \in \mathbb{R}^{1 \times d_v}$
- We have the previous keys and values cached:
 - $K_{1:t-1} \in \mathbb{R}^{(t-1) \times d_k}$
 - $V_{1:t-1} \in \mathbb{R}^{(t-1) \times d_v}$
- We append k_t to $K_{1:t-1}$ and v_t to $V_{1:t-1}$ and compute attention:

$$\bullet \quad z_t = \text{softmax} \left(\frac{q_t K_{1:t}^T}{\sqrt{d_k}} \right) V_{1:t}$$

Without caching,
we recompute:

$$\begin{aligned} K_{1:t} &= [h_1; h_2; \dots; h_t] W_k \\ V_{1:t} &= [h_1; h_2; \dots; h_t] W_v \end{aligned}$$

Key value cache

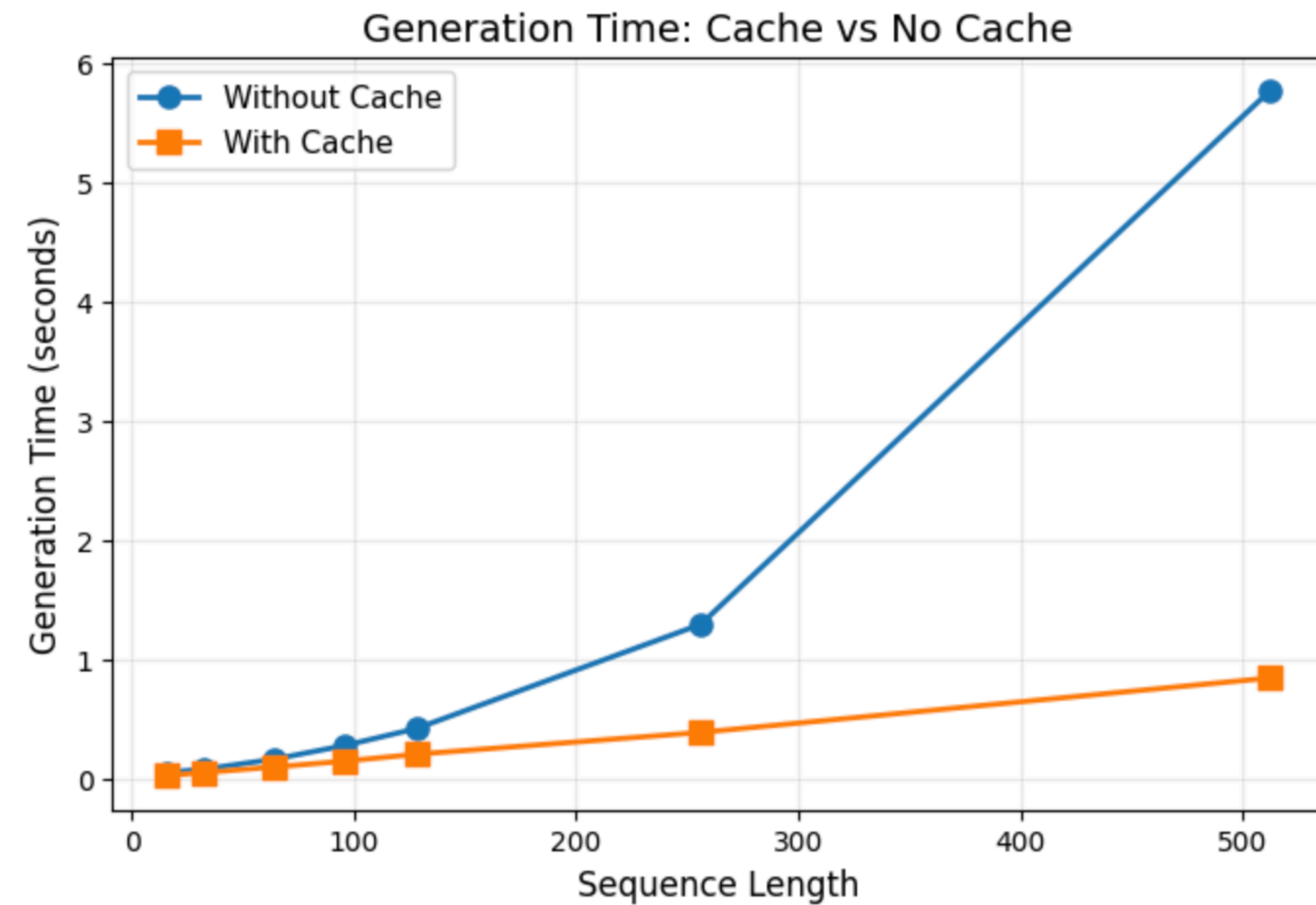


Code example

```
if use_cache and self.cache_k is not None:
    # Only compute K, V for the new token(s)
    K_new = self.k_proj(x_norm)
    V_new = self.v_proj(x_norm)

    # Append to cache
    K = torch.cat([self.cache_k, K_new], dim=1)
    V = torch.cat([self.cache_v, V_new], dim=1)

    # Update cache
    self.cache_k = K
    self.cache_v = V
```



Why is decoding slow?

- **Memory:** transferring data (weights, activations)
- **Compute:** performing computations

$$\text{time} = \max \left(\frac{\text{OperationFLOPs}}{\text{DeviceFLOP/s}}, \frac{\text{DataTransferred(GB)}}{\text{MemoryBandwidth(GB/s)}} \right)$$

“Compute-bound”

e.g. $A=BC$: $O(mkn)$ FLOPs,
 $O(mk+kn)$ bytes

“Memory-bound”

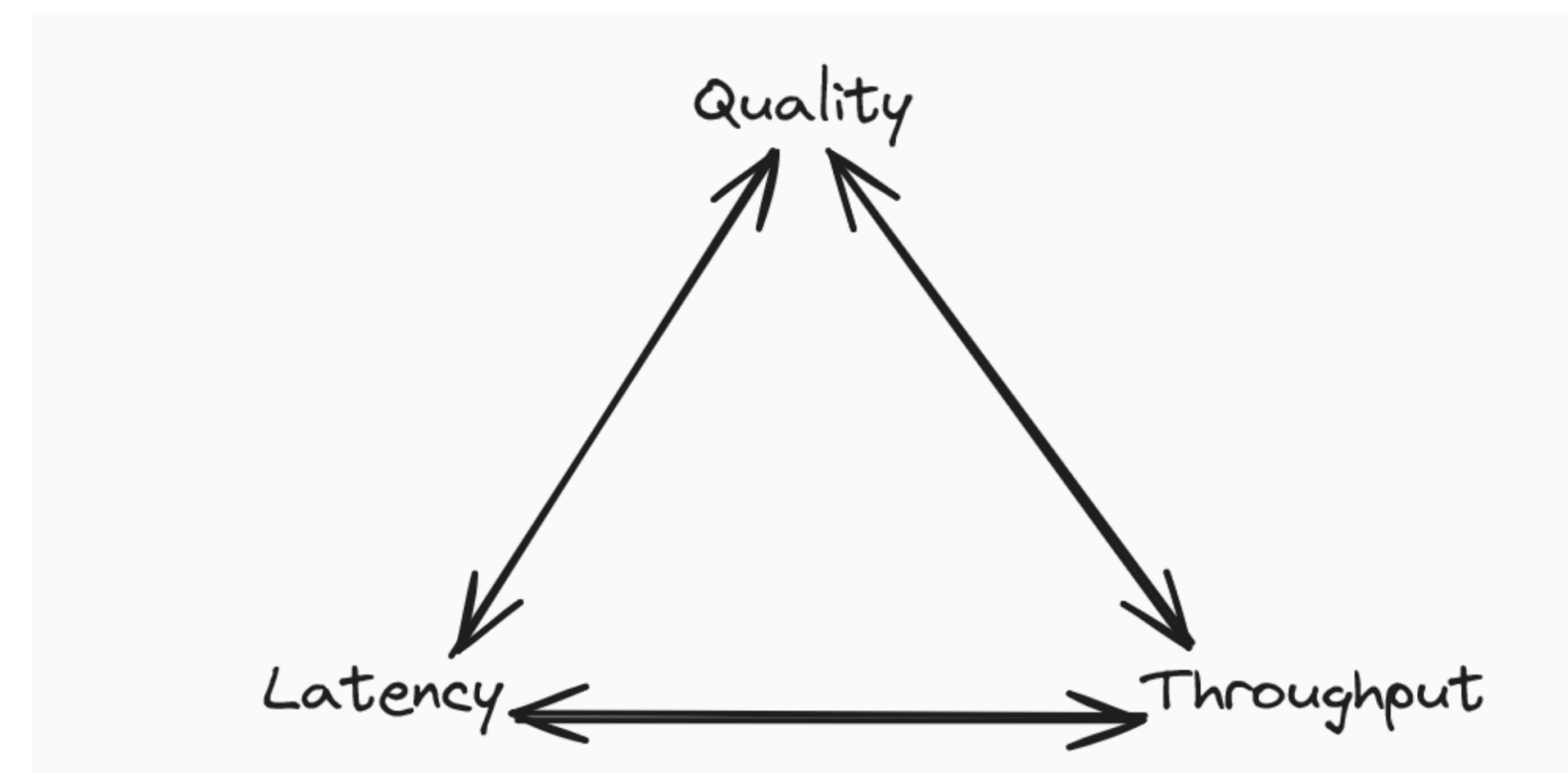
e.g. $a=Bx$: $O(mk)$ FLOPs
 $O(mk)$ bytes

Example: decoding one token

What does speeding up mean?

- **Latency**: how long does a user wait for a response?
 - Time to first token, time per request
- **Throughput**: how many requests are completed per second?
 - Tokens per second, requests per second

Next: a very quick tour of strategies



Speeding up a single token

Goal	Strategy	Examples
Reduce memory bandwidth (shrink data to move)	Quantization, distillation, architecture elements	GPTQ, AWQ, GQA/MQA
Increase FLOP/s	Optimize how operations run on given hardware	FlashAttention, torch.compile/kernel fusion
Decrease FLOPs	Reduce FLOPs in the architecture	Mixture-of-Experts, Mamba

$$\text{time} = \max \left(\frac{\text{OperationFLOPs}}{\text{DeviceFLOP/s}}, \frac{\text{DataTransferred(GB)}}{\text{MemoryBandwidth(GB/s)}} \right)$$

Speeding up a full sequence

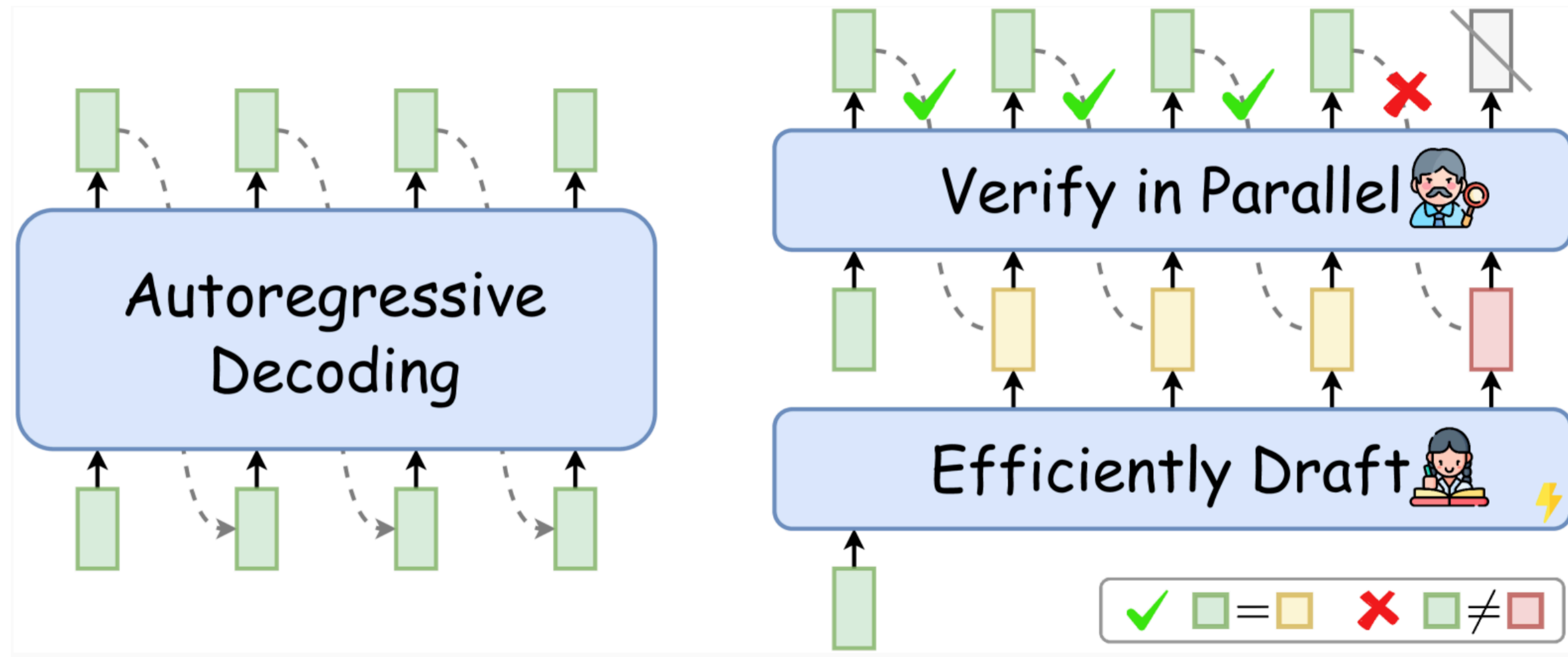
Strategy	Idea	Examples
Parallelize over time	Draft multiple tokens cheaply, verify in parallel with the target model	Speculative decoding
Parallelize over time	Generate multiple tokens in parallel	Non-autoregressive models, e.g. diffusion

Speculative decoding

- Some tokens are easier to predict than others
 - The cow jumped over the moon . <EOS>
- Idea:
 - Use a small **draft** model q to generate a few tokens ahead
 - Larger model p processes the generated tokens all at once
 - Accept drafted tokens with probability:

$$\alpha_i = \min \left(1, \frac{p(\text{token} | \dots)}{q(\text{token} | \dots)} \right)$$

Speculative decoding



Speeding up multiple sequences

will cover these in more detail in future lecture

Strategy	Idea	Examples
State re-use	Shared prefixes => shared KV cache	PagedAttention, RadixAttention (SGLang)
Improved batching	Better scheduling of concurrent generations	Continuous batching
Program-level optimization	Optimize the computation graph of the full generation program	SGLang, DSPy

Advanced tokenization methods

In week 1, we covered byte-pair encoding (BPE).

Suppose the word “pingaprong” never occurred in the training data used to build the vocab. How would this get encoded with BPE?

- A. It would get mapped to a special `_UNKNOWN` token.
- B. It will get tokenized as one token per UTF-8 byte.
- C. It will get greedily broken into the longest pieces which *are* tokens.

In week 1, we covered byte-pair encoding (BPE).

Suppose the word “pingaprong” never occurred in the training data used to build the vocab. How would this get encoded with BPE?

- A. It would get mapped to a special `_UNKNOWN` token.
- B. It will get tokenized as one token per UTF-8 byte.
- C. ☒ It will get greedily broken into the longest pieces which *are* tokens.

pingaprong

How would you want your tokenizer to tokenize the white space in this Python code?

```
from heapq import merge
```

```
def merge_sort(m):  
    if len(m) <= 1:  
        return m
```

```
    middle = len(m) // 2  
    left = m[:middle]  
    right = m[middle:]
```

```
    left = merge_sort(left)  
    right = merge_sort(right)  
    return list(merge(left, right))
```

Tokenizing whitespace

T5
(n=85)

```
fromheapqimportmergedefmerge_sort(m):iflen(m)<unk>=1  
:returnmmiddle=len(m)//2left=m[:middle]right=m[mi  
ddle:]left=merge_sort(left)right=merge_sort(right)  
returnlist(merge(left,right))
```

Tokenizing whitespace

GPT-3

(n=107)

```
from heapq import merge
```

```
def merge_sort(m):  
    if len(m) <= 1:  
        return m
```

```
    middle = len(m) // 2  
    left = m[:middle]  
    right = m[middle:]
```

```
    left = merge_sort(left)  
    right = merge_sort(right)  
    return list(merge(left, right))
```

Tokenizing whitespace

GPT-5

(n=70)

```
from heapq import merge
```

```
def merge_sort(m):  
    if len(m) <= 1:  
        return m
```

```
    middle = len(m) // 2  
    left = m[:middle]  
    right = m[middle:]
```

```
    left = merge_sort(left)  
    right = merge_sort(right)  
    return list(merge(left, right))
```

Handling multilinguality

- The encoder-decoder solution: learn a separate vocabulary for each language, use separate vocab embedding matrices in the encoder and decoder.
- Bad solution: learn a single vocabulary from whatever data you have on hand.
- Better solution: deliberate language balancing.

Evaluating tokenizers

- Fertility
 - the average number of tokens it takes to represent text x divided by the total number of words or characters in text x .
 - similar to compression ratio

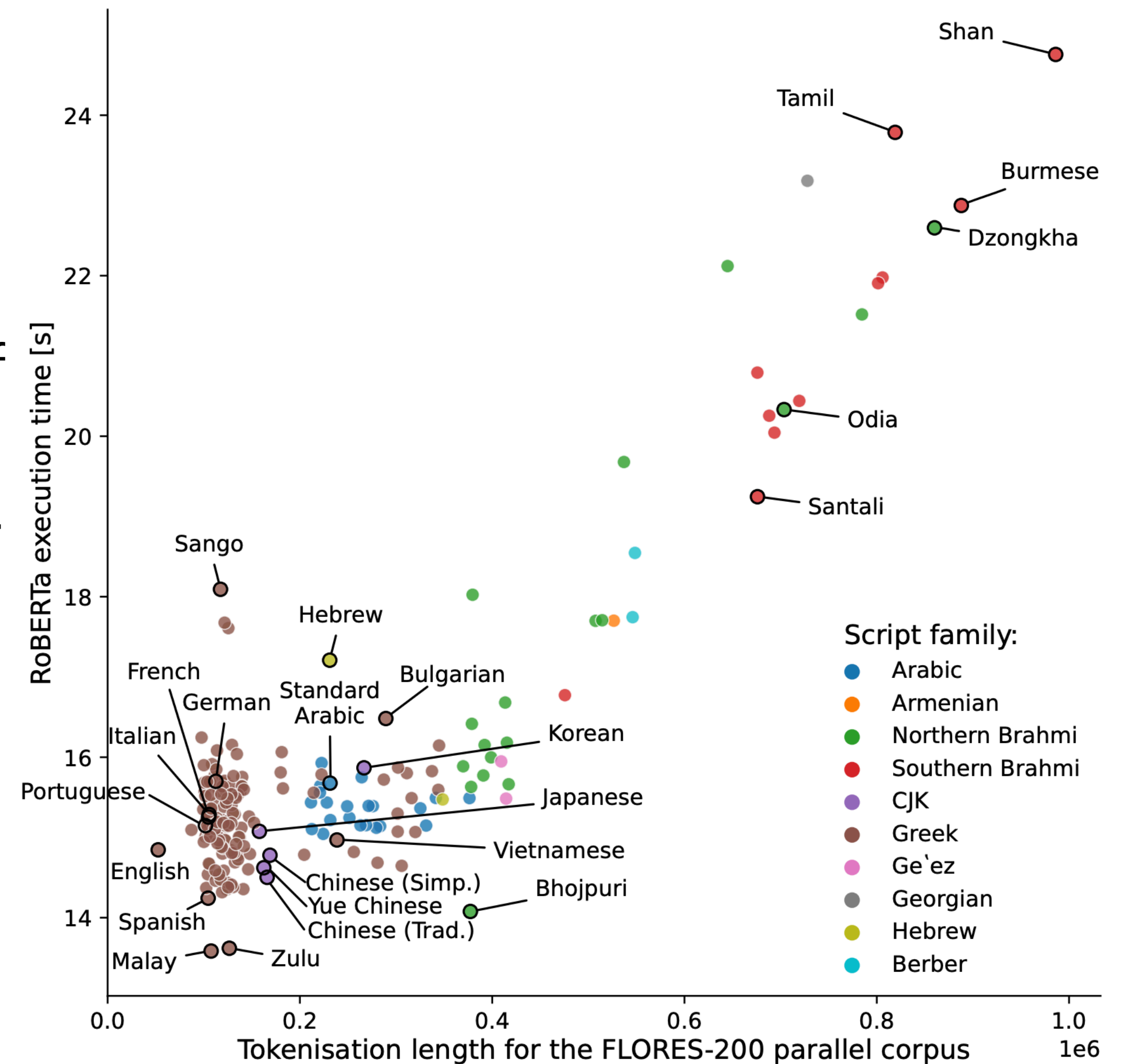
Evaluating tokenizers

- Parity
 - how fairly a tokenizer treats equivalent sentences in different languages
 - parity between languages A and B is 1 if on average equivalent sentences in the two languages map to the same number of tokens

	Parity with respect to English		
	GPT-2 RoBERTa	ChatGPT GPT-4	FlanT5
Bulgarian	5.51	2.64	—
Burmese	16.89	11.70	—
Chinese (Simplified)	3.21	1.91	—
Dzongkha	16.36	12.33	—
English	1.00	1.00	1.00
French	2.00	1.60	1.60
German	2.14	1.58	1.37
Italian	2.01	1.64	2.18
Japanese	3.00	2.30	—
Jingpho	2.65	2.35	3.41
Maori	2.45	2.35	3.28
Norwegian Bokmål	1.86	1.56	2.24
Odia	13.38	12.48	—
Pangasinan	1.66	1.57	2.18
Portuguese	1.94	1.48	2.21
Romanian	2.48	1.88	1.50
Santali	12.86	12.80	—
Shan	18.76	15.05	—
Spanish	1.99	1.55	2.23
Standard Arabic	4.40	3.04	—
Tumbuka	2.78	2.57	3.29
Vietnamese	4.54	2.45	—

Evaluating tokenizers

- Parity
 - how fairly a tokenizer treats equivalent sentences in different languages
 - parity between languages A and B is 1 if on average equivalent sentences in the two languages map to the same number of tokens



Why not just use byte- or character-level
tokenization?

Why not just use byte- or character-level tokenization?

- Cons:
 - Extremely long sequence lengths
 - Individual characters/bytes don't carry much semantic meaning

However, some attempts have been made.

- Newer model architectures (e.g. state space models such as Mamba) eliminate the $O(n^2)$ parameter growth with sequence length n . This makes them more amenable to byte-level tokenization.
- MambaByte: Token-free Selective State Space Model
- Other approaches try to modify existing Transformer architectures to make them more amenable to byte-level tokens.
- ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models

Why do we insist on token boundaries at
whitespace boundaries?

Why do we insist on token boundaries at whitespace boundaries?

- **Pretokenization:** the method used to break strings of text into initial chunks (which get fed into the BPE or other algorithm)
 - For English, this is typically whitespace boundaries, punctuation, numbers etc. For Chinese, there are no whitespace boundaries to split on.
 -

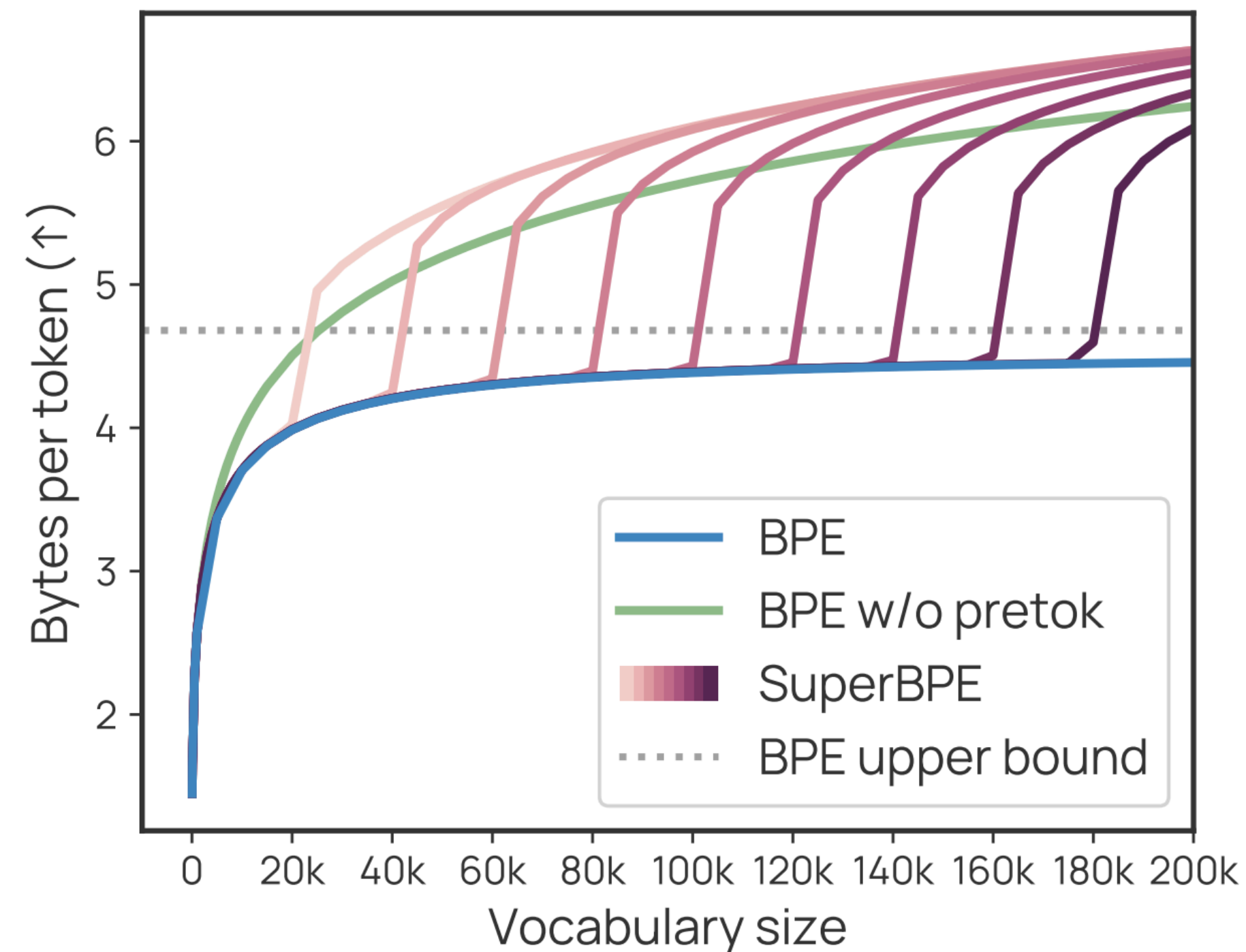
Why do we insist on token boundaries at whitespace boundaries?

- **Pretokenization:** the method used to break strings of text into initial chunks (which get fed into the BPE or other algorithm)
 - For English, this is typically whitespace boundaries, punctuation, numbers etc. For Chinese, there are no whitespace boundaries to split on.
- Alternative:
 - don't include whitespace in the pretokenization stage.

Tokenization that transcends whitespace

BPE: By the way, I am a fan of the Milky Way.

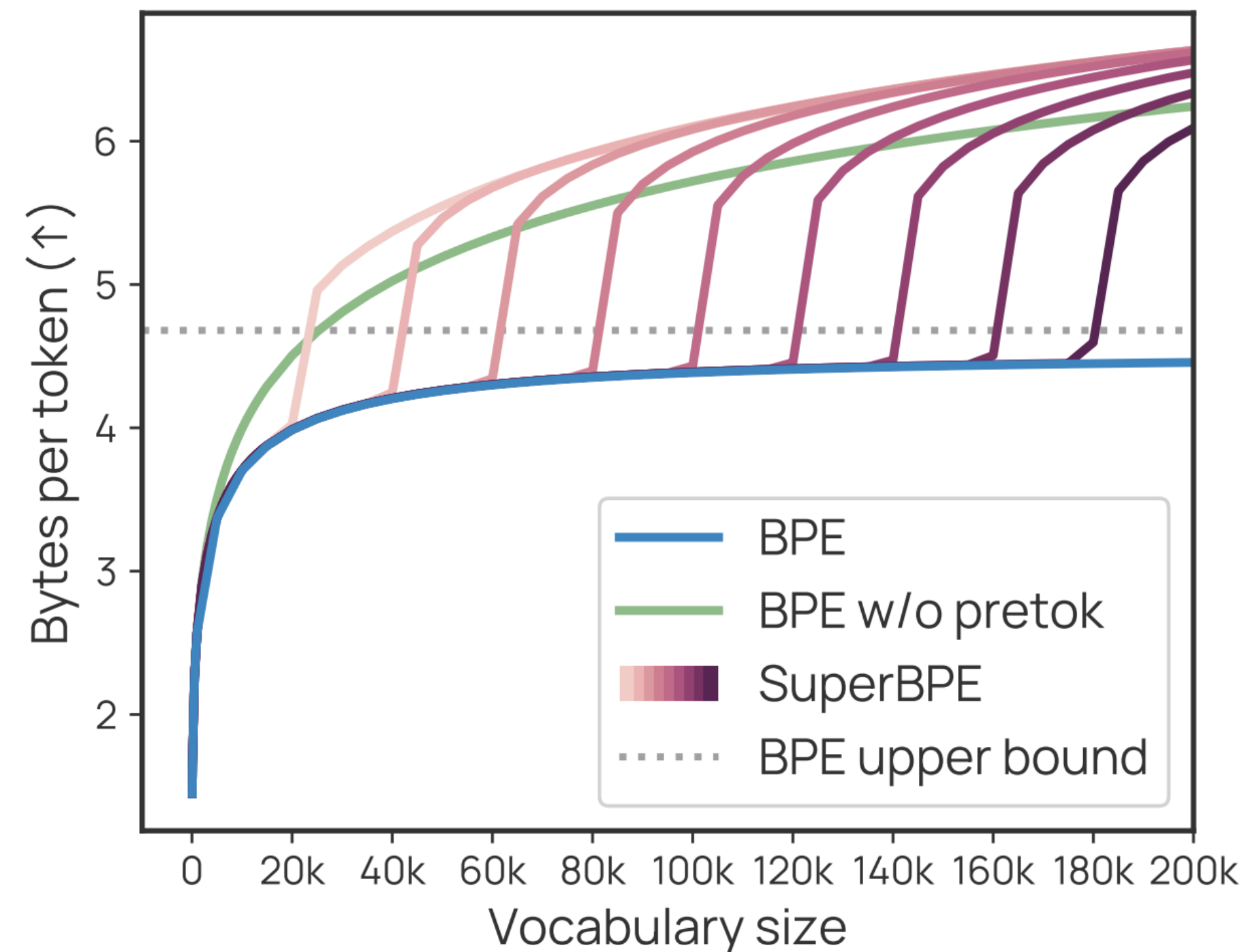
SuperBPE: By the way, I am a fan of the Milky Way.



Tokenization that transcends whitespace

BPE: By the way, I am a fan of the Milky Way.

SuperBPE: By the way, I am a fan of the Milky Way.



SuperBPE tokenization

1. Pretokenize the text using standard rules.
2. Start running the regular BPE algorithm (that we covered in week 1)
3. Once the working vocabulary reaches size t , start ignoring the pretokenization—tokens can now be merged across whitespace boundaries.

SuperBPE tokenization

1. Pretokenize the text using standard rules.
2. Start running the regular BPE algorithm (that we covered in week 1)
3. Once the working vocabulary reaches size t , start ignoring the pretokenization—tokens can now be merged across whitespace boundaries.

Intuition: tokenizer first learn base units of semantic meaning, then, after the transition point, it learns to combine these units into common sequences for a much more efficient vocabulary.

SuperBPE tokenization

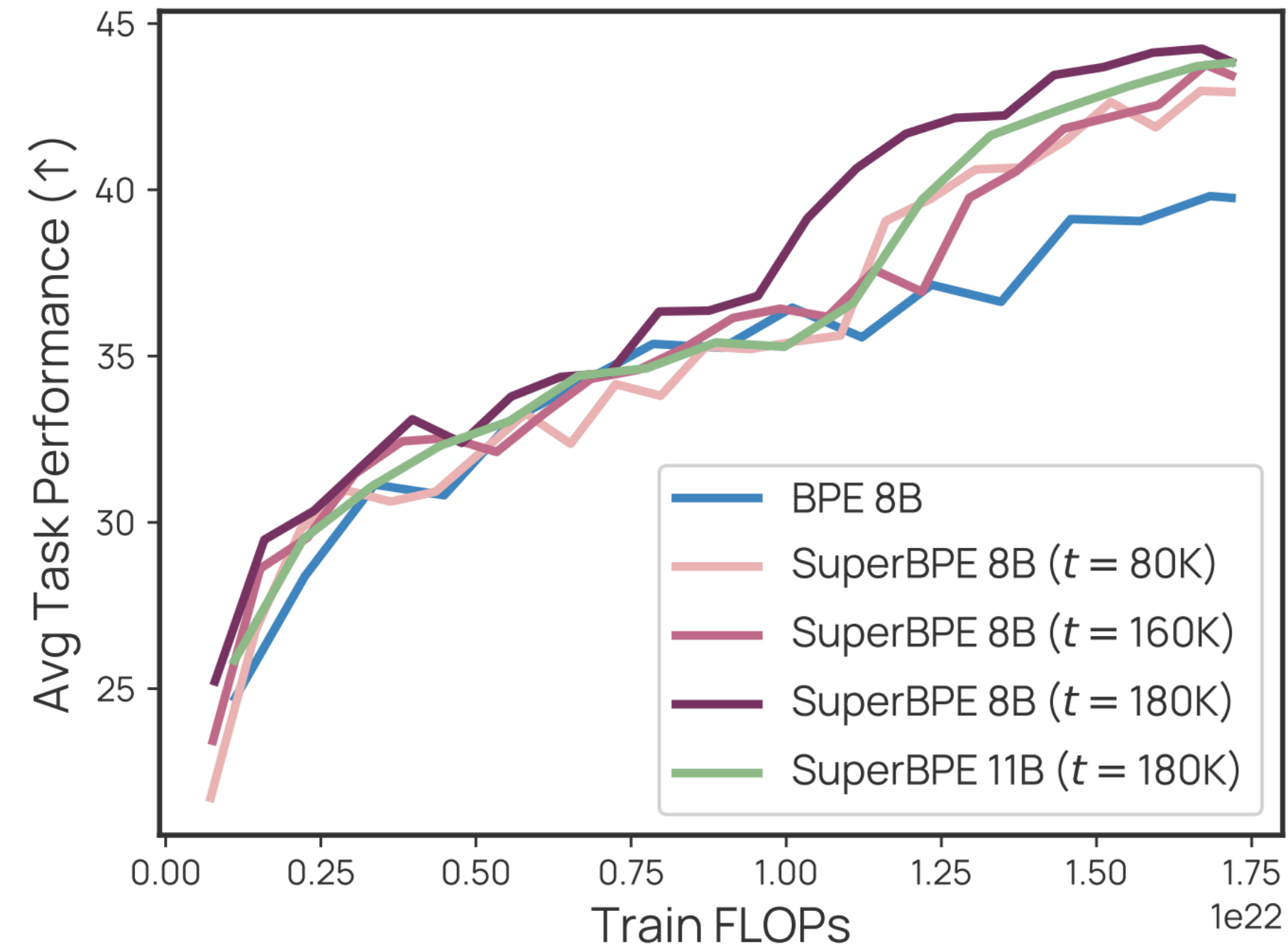


Figure 3: **Average task performance on 30 downstream tasks, evaluated at every 5000 steps in model pretraining.** We see that SuperBPE models consistently outperform the baseline that uses a BPE tokenizer. All compared models share the same vocabulary size and train budget; t denotes the transition point in SuperBPE’s pretokenization curriculum.

Thanks!