Decoding from Neural Text Generation Models

Antoine Bosselut

Generation Model Basics

1. At each time step, model computes a score o_n for each token in our vocabulary, $w_n \in V$

$$O_n = f(\{y\}_{<t})$$

$f(\,.\,)$ is your model

Generation Model Basics

1. At each time step, model computes a score o_n for each token in our vocabulary, $w_n \in V$

$$O_n = f(\{y\}_{<t})$$

$f(\,.\,)$ is your model

2. Compute a probability distribution over these scores (usually softmax)

$P(\,.\,)$ is your distribution over tokens

$$P(y_t = w_n \mid \{y\}_{<t}) = \frac{e^{o_n}}{\sum_{m=1}^{M} e^{o_m}}$$

Generation Model Basics

1. At each time step, model computes a score o_n for each token in our vocabulary, $w_n \in V$

$$O_n = f(\{y\}_{<t})$$

$f(\,.\,)$ is your model

2. Compute a probability distribution over these scores (usually softmax)

$P(\,.\,)$ is your distribution over tokens

$$P(y_t = w_n \mid \{y\}_{<t}) = \frac{e^{o_n}}{\sum_{m=1}^{M} e^{o_m}}$$

3. Define a function to select a token from this distribution

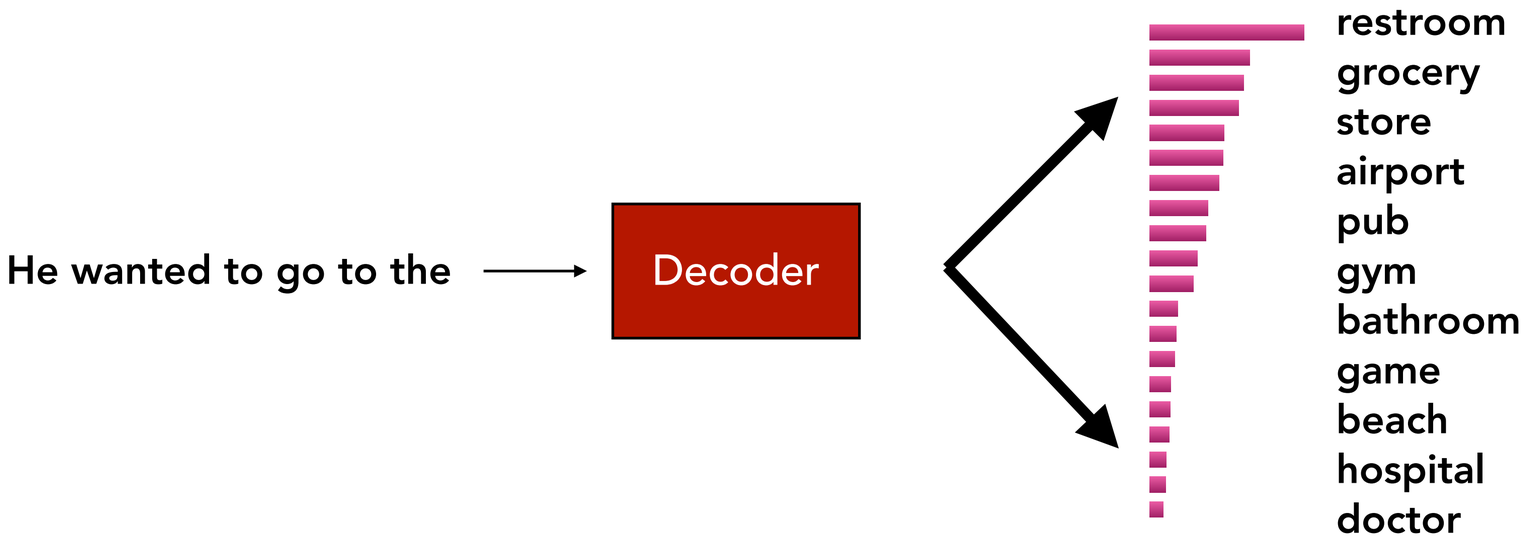
$$\hat{y}_t = g(P(y_t \mid \{y\}_{<t}))$$

$g(\,.\,)$ is your decoding algorithm

Simplest approach: Argmax Decoding

- g = select the token with the highest probability:

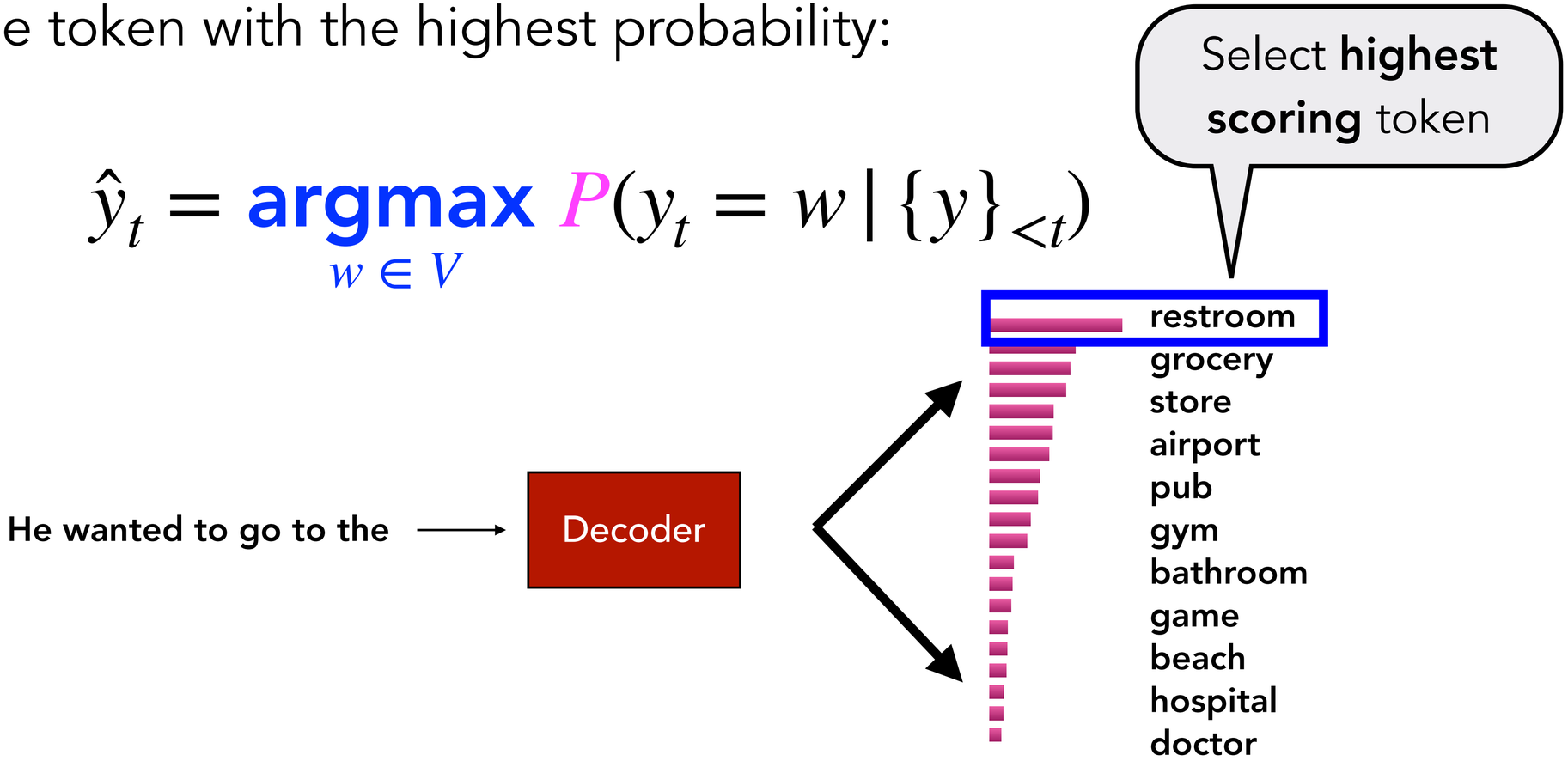
$$\hat{y}_t = \underset{w \in V}{\textbf{argmax}}\ P(y_t = w \mid \{y\}_{<t})$$



Simplest approach: Argmax Decoding

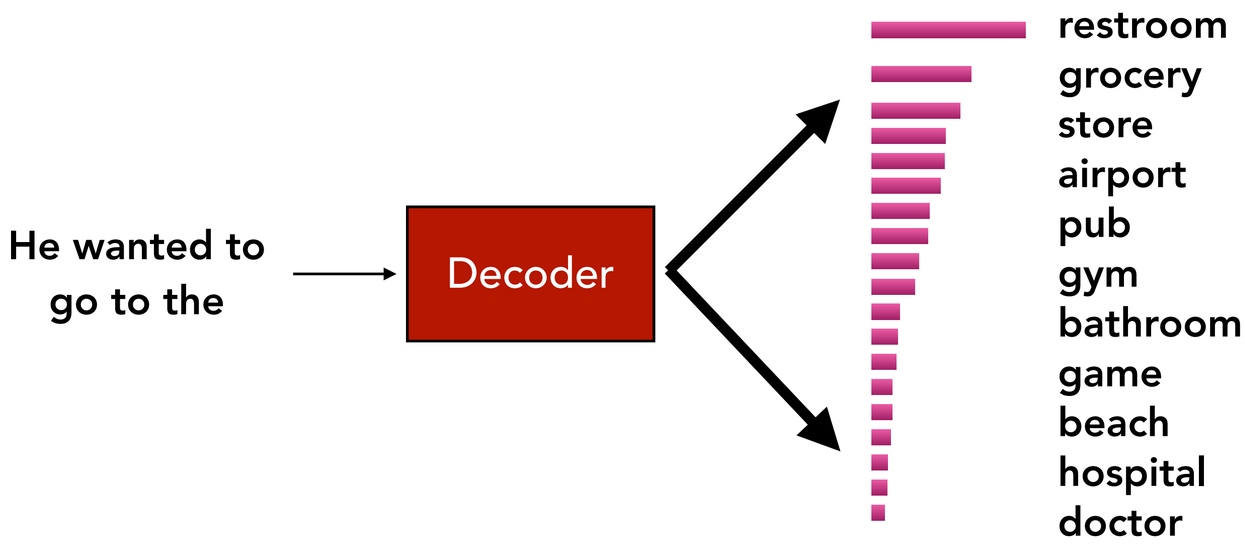
- g = select the token with the highest probability:

$$\hat{y}_t = \underset{w \in V}{\textbf{argmax}}\ P(y_t = w \mid \{y\}_{<t})$$



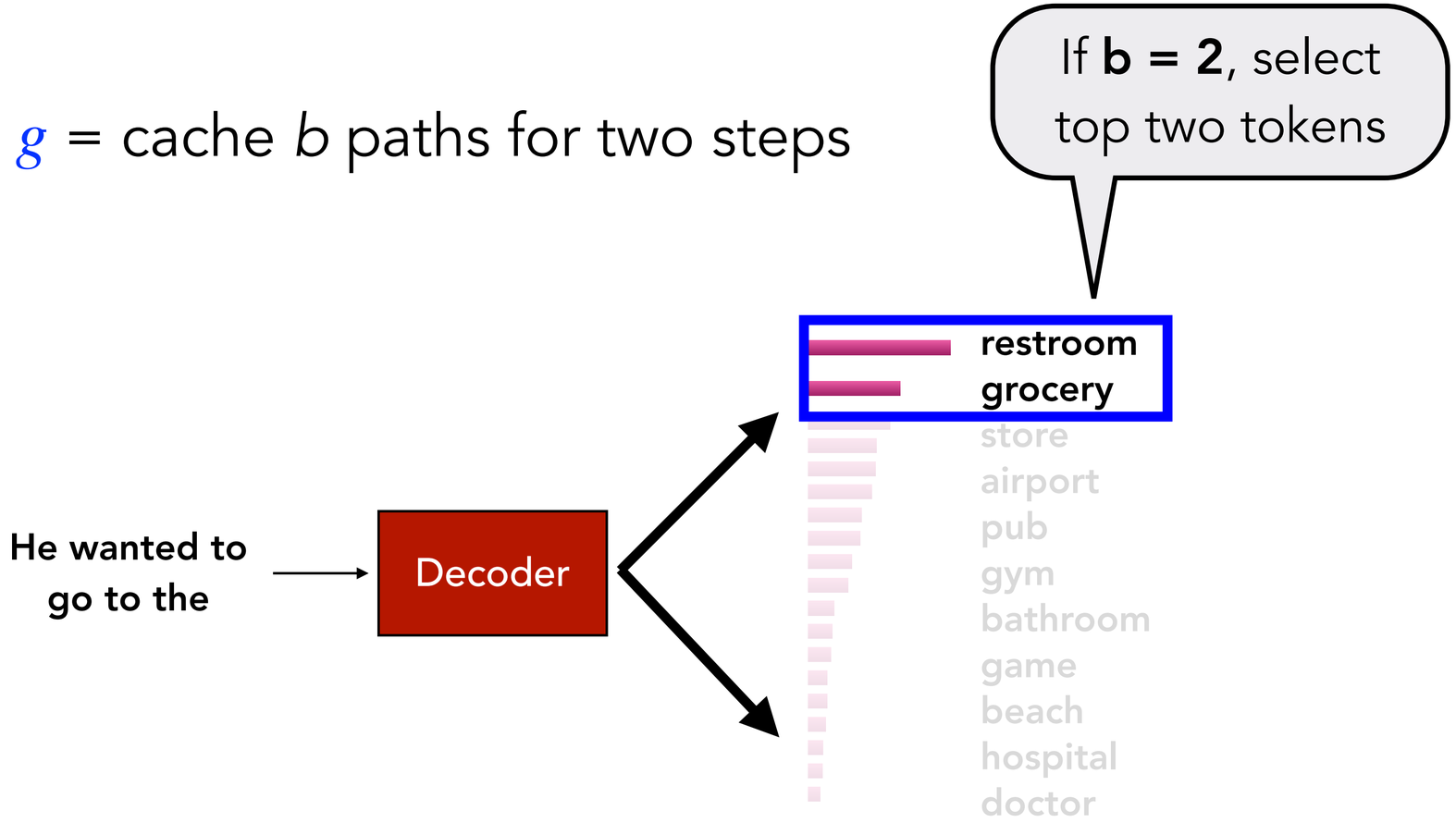
Maybe we need more options: Beam Search

- g = cache b paths for two steps



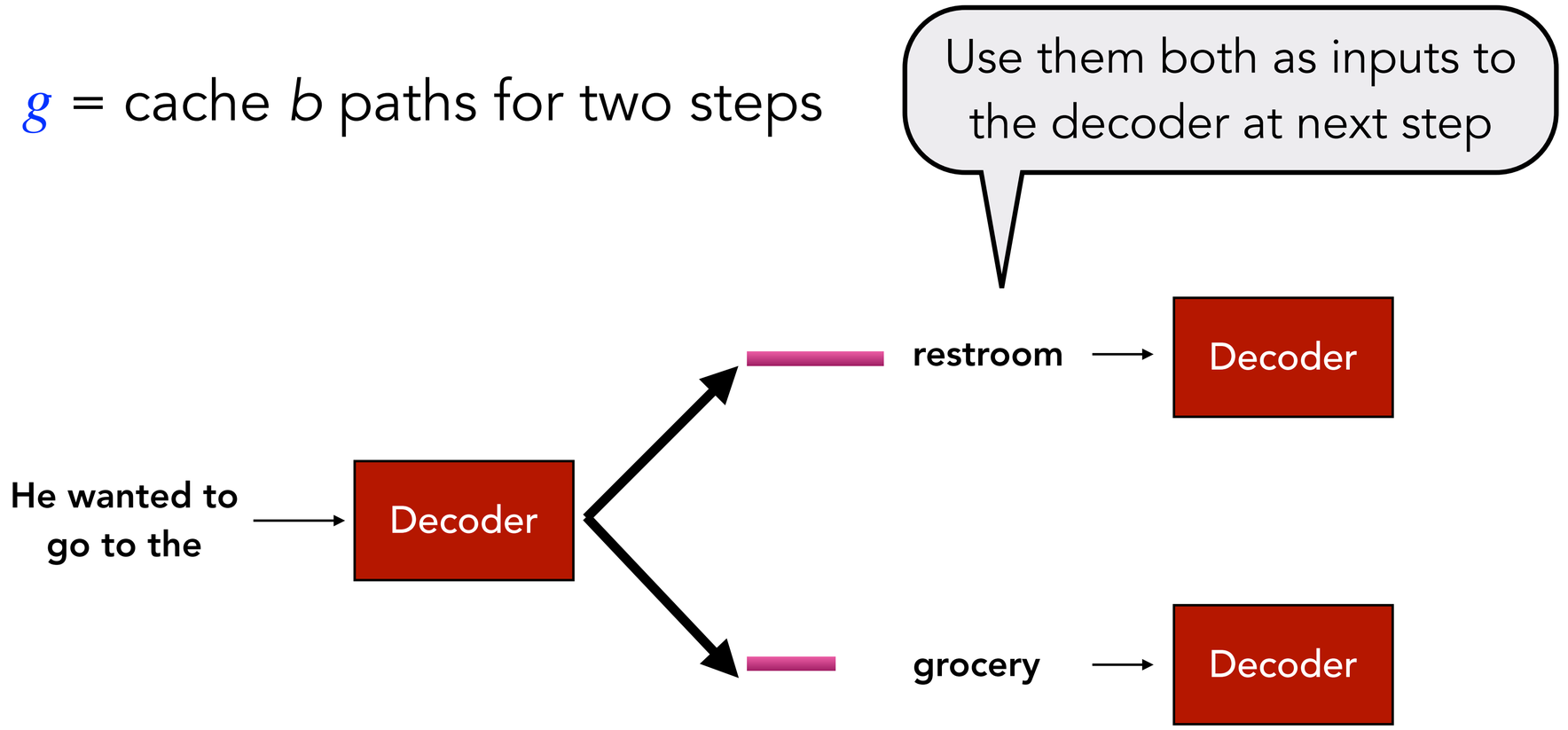
Maybe we need more options: Beam Search

- g = cache b paths for two steps



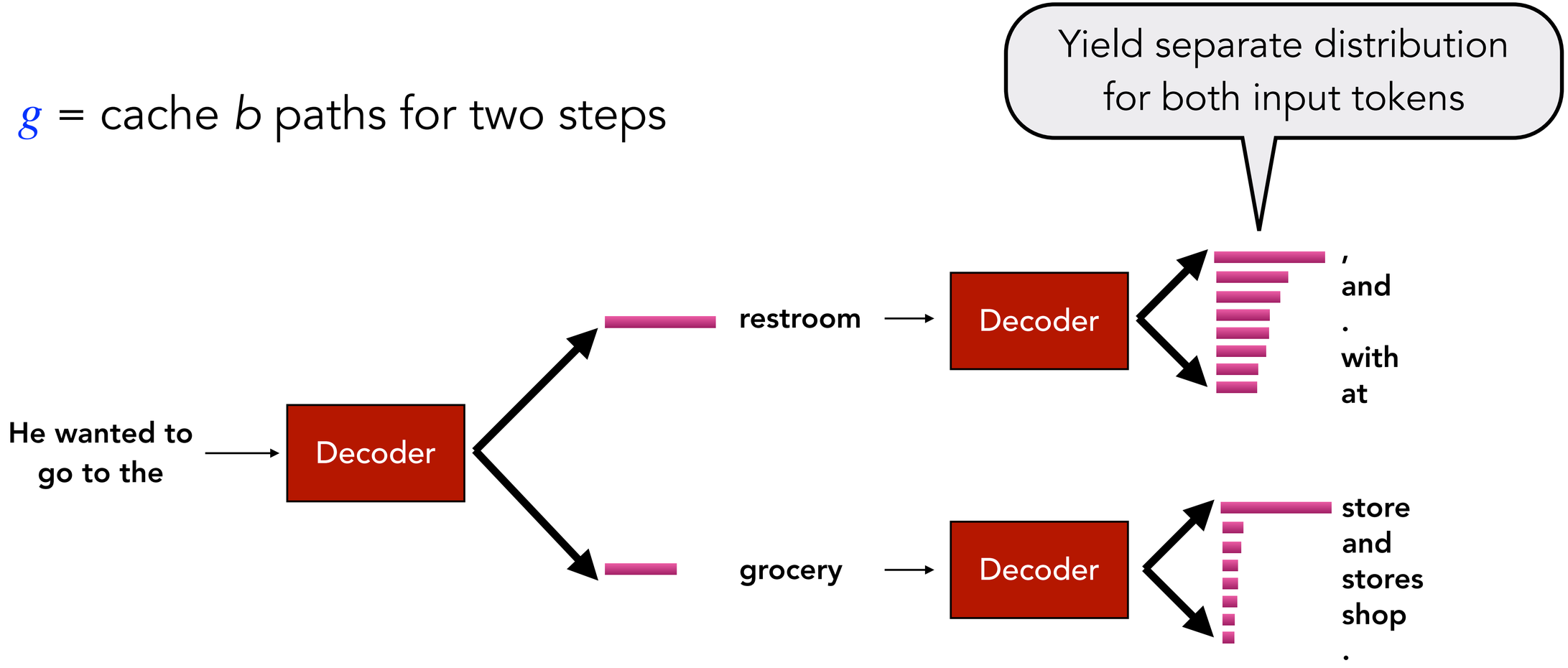
Maybe we need more options: Beam Search

- g = cache b paths for two steps



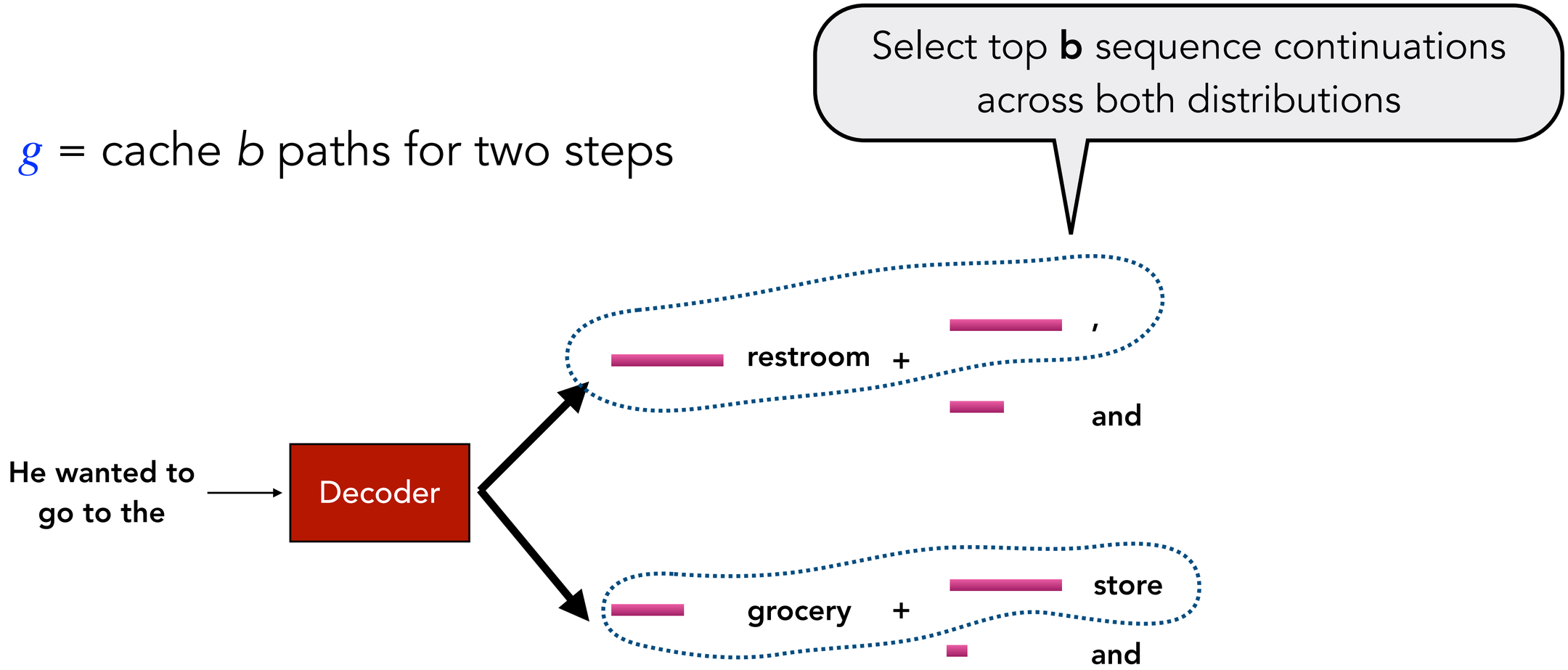
Maybe we need more options: Beam Search

- g = cache b paths for two steps



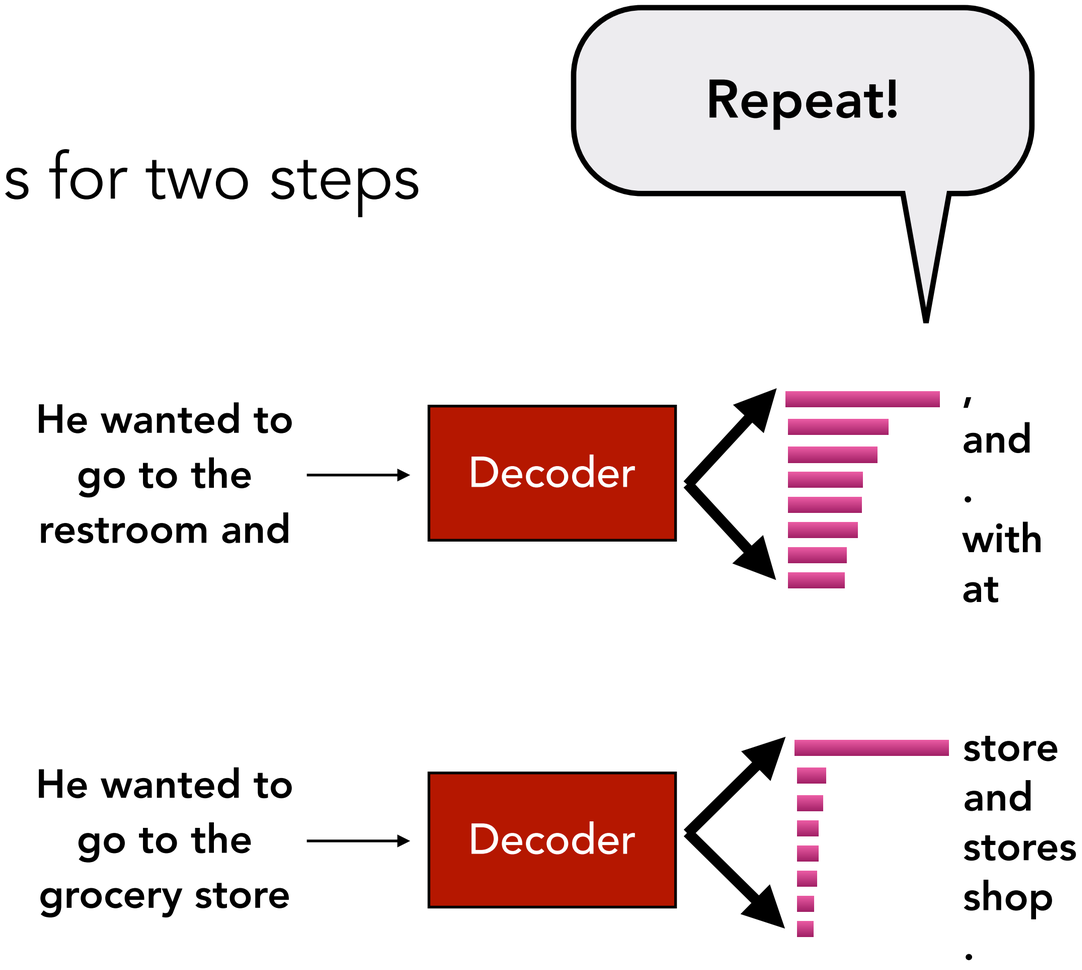
Maybe we need more options: Beam Search

- g = cache b paths for two steps



Maybe we need more options: Beam Search

- g = cache b paths for two steps



Does this penalize longer sequences?

$$s(Y) = \sum_{t=1}^{T} \log P\left(y_t \mid \{y\}_{<t}\right)$$

Shorter sequences will score better!

Does this penalize longer sequences?

- **Solution**: Normalize by token length of sequence

$$s(Y) = \frac{1}{T} \sum_{t=1}^{T} \log P\left(y_t \mid \{y\}_{<t}\right)$$

- **Solution:** Normalize by token length relative to reference sequence

$$s(Y) = \frac{1}{lp(Y)} \sum_{t=1}^{T} \log P\left(y_t \mid \{y\}_{<t}\right) \qquad lp(Y) = \frac{(5 + |Y|)^{\alpha}}{(5 + 1)^{\alpha}}$$

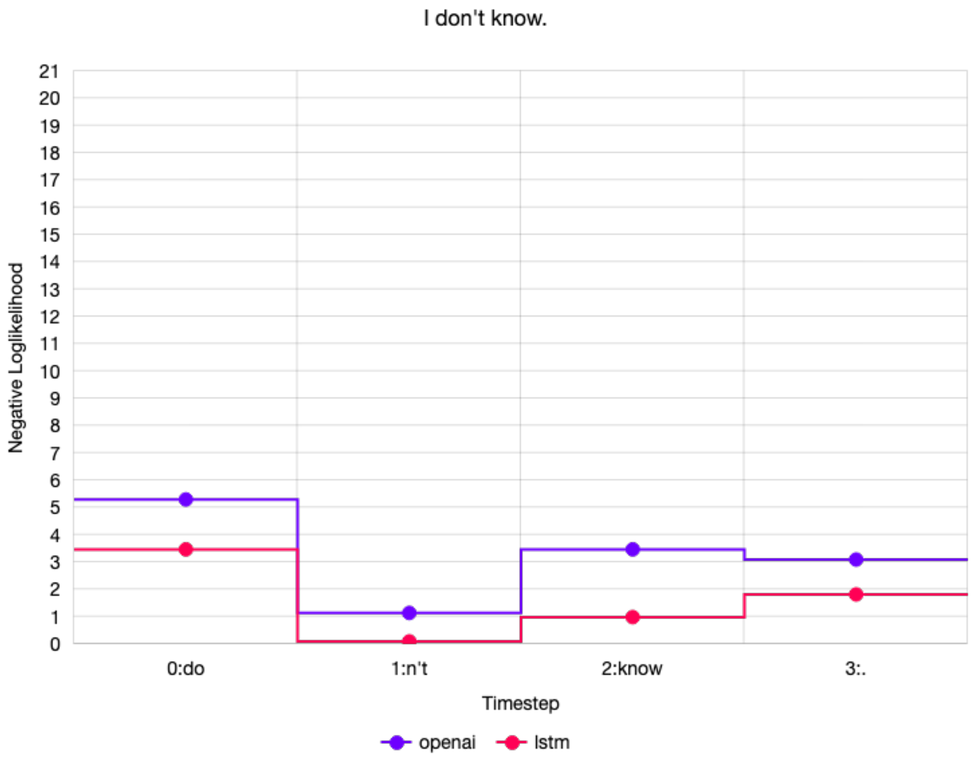
Beam search gets repetitive and repetitive

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Continuation: The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the **Universidad Nacional Autónoma de México (UNAM)** and **the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México...**

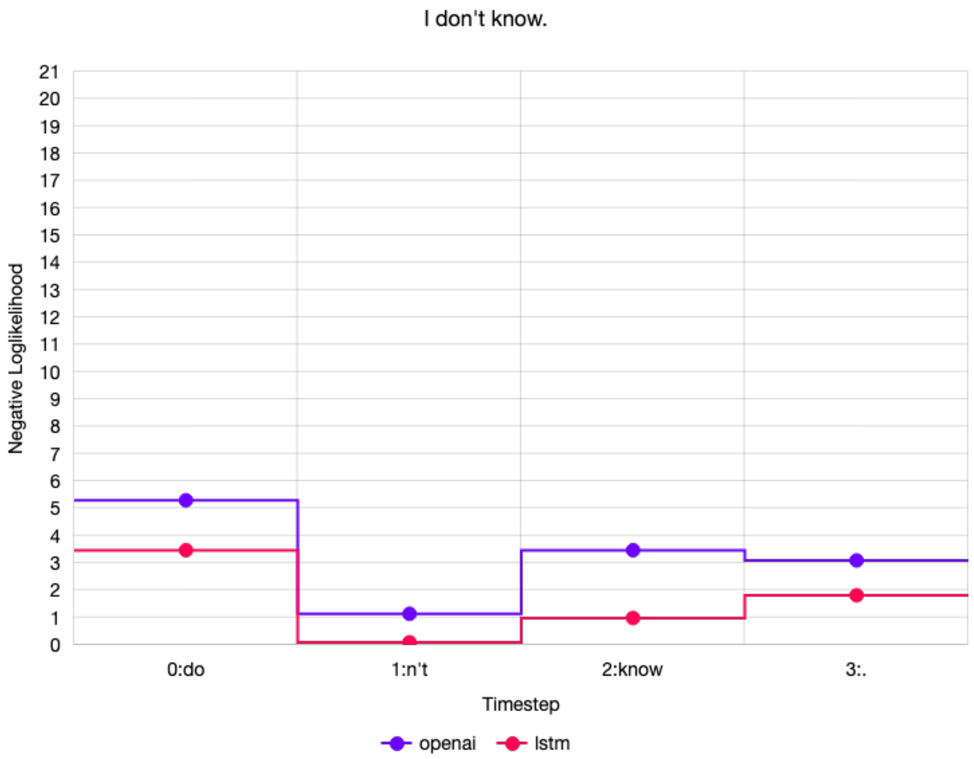
Holtzman et. al., ICLR 2020

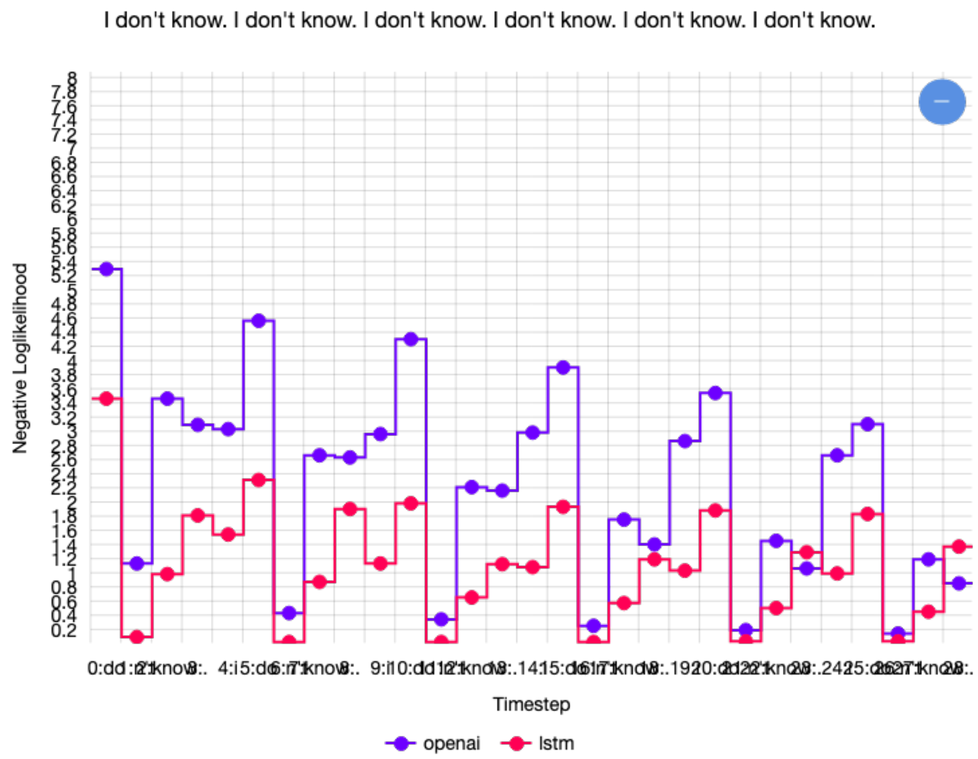
Why does this happen?



Holtzman et. al., ICLR 2020

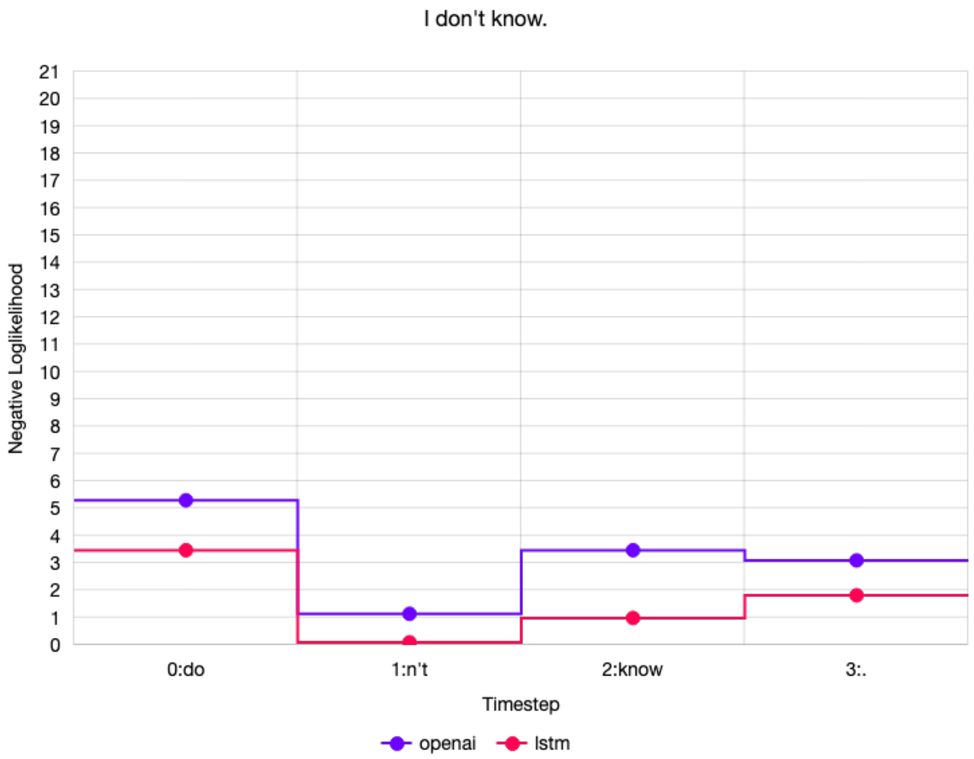
Artifact of Maximum Likelihood Training

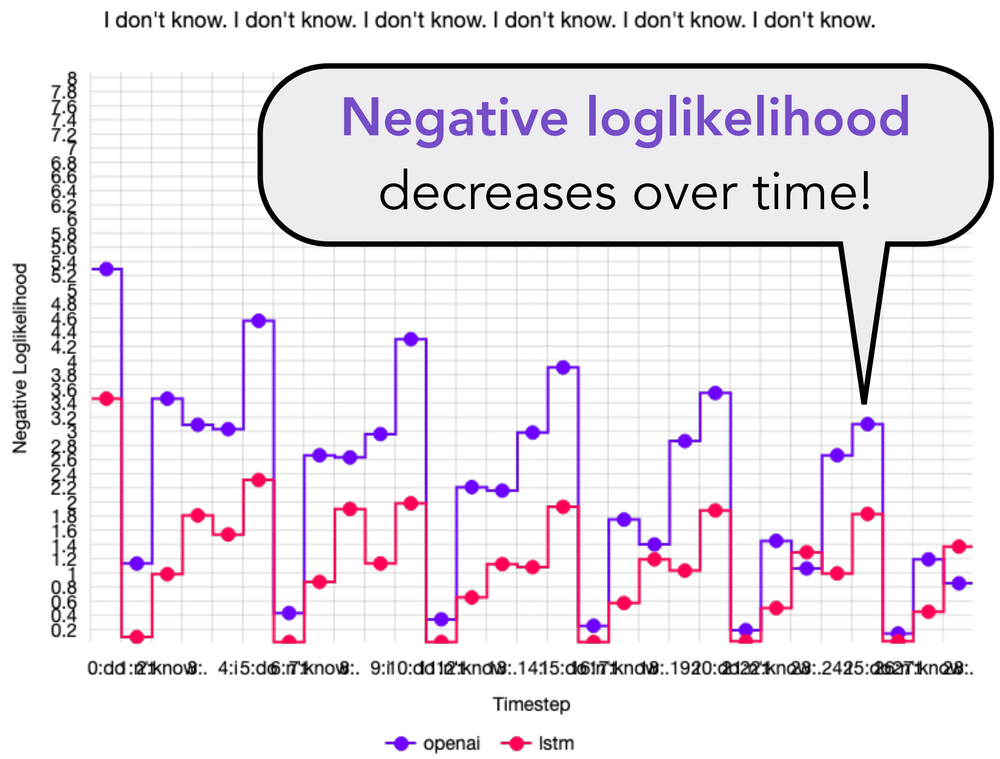




Holtzman et. al., ICLR 2020

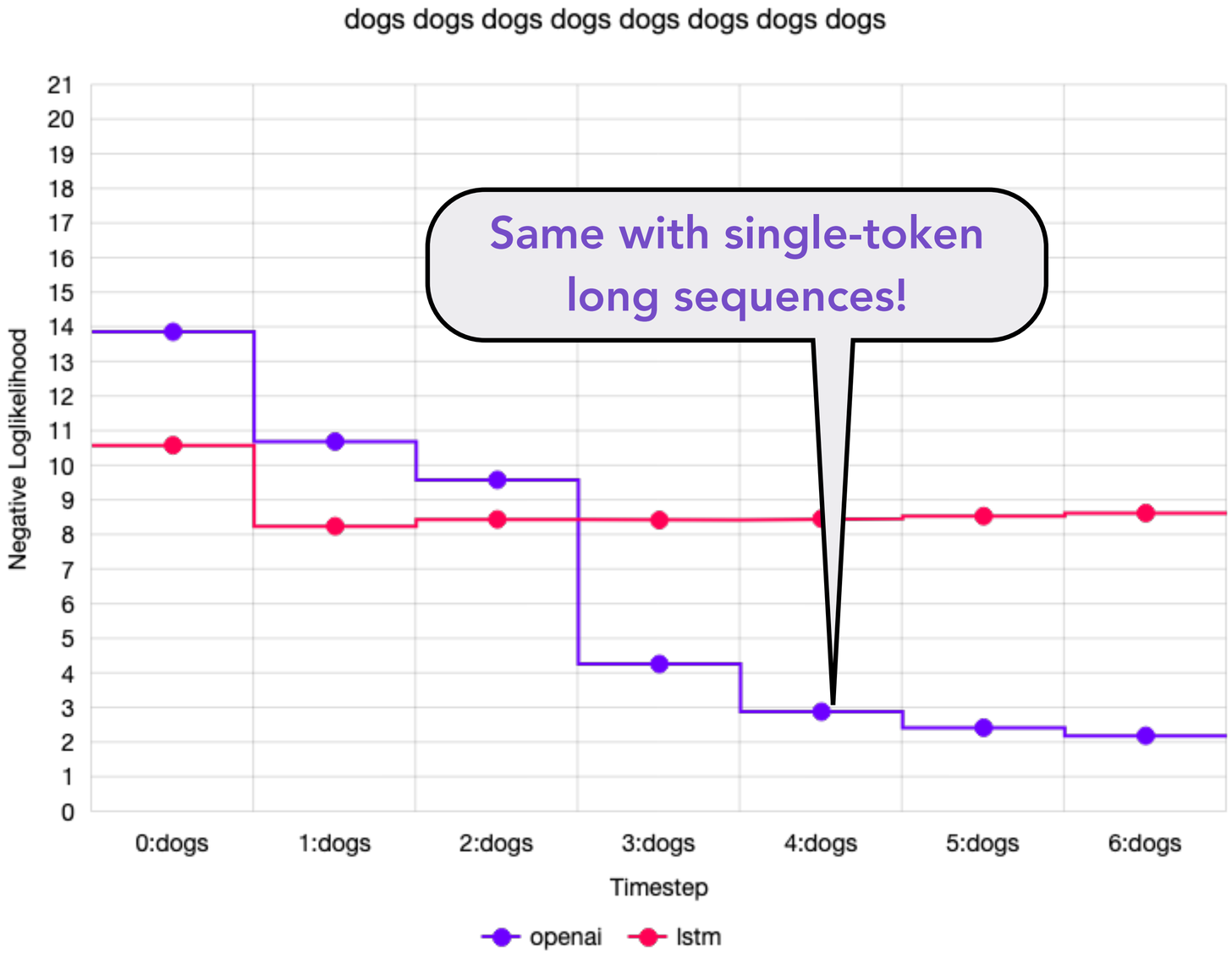
Artifact of Maximum Likelihood Training





Holtzman et. al., ICLR 2020

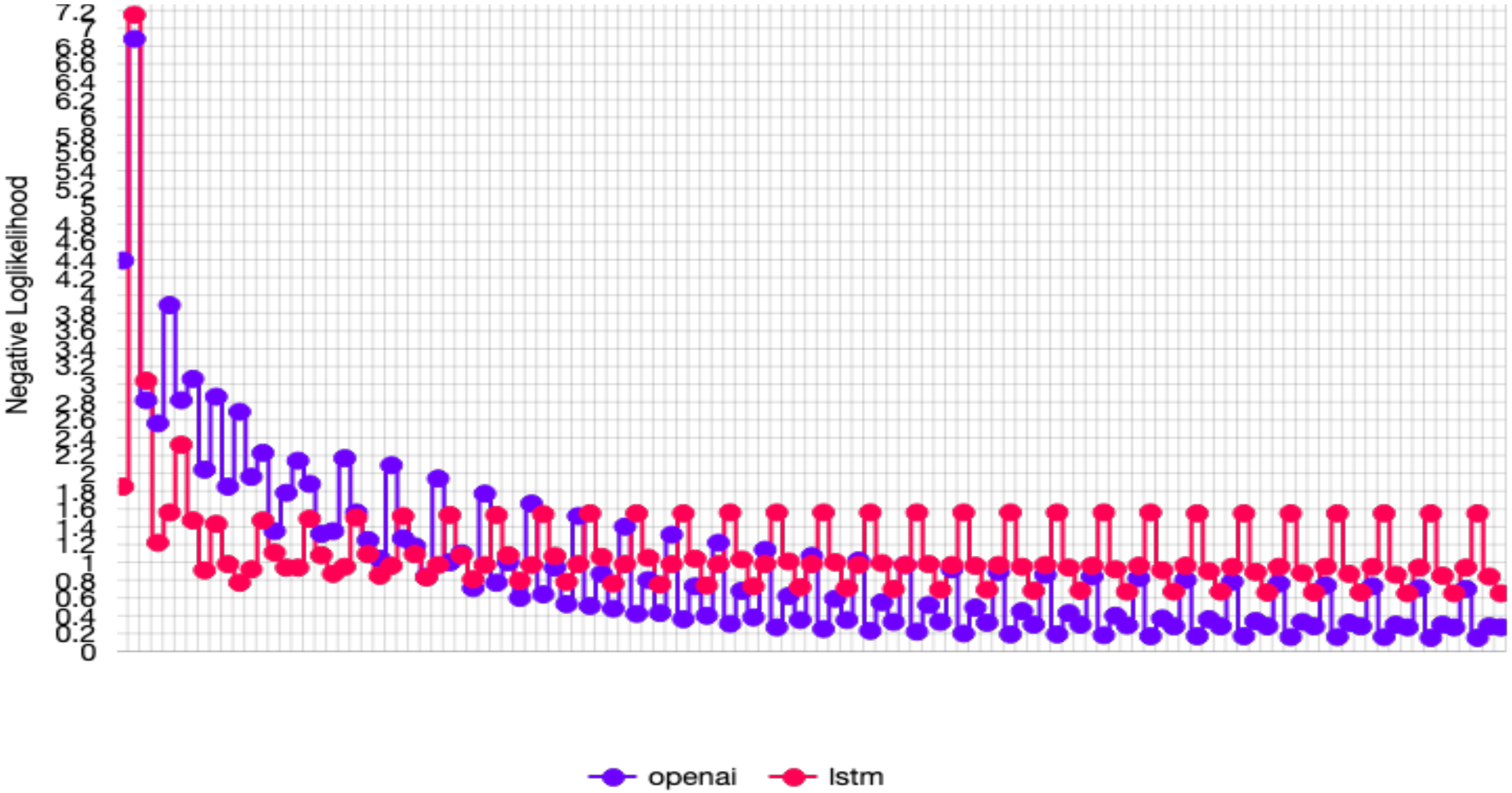
Beam search gets repetitive and repetitive



Holtzman et. al., ICLR 2020

Beam search gets repetitive and repetitive

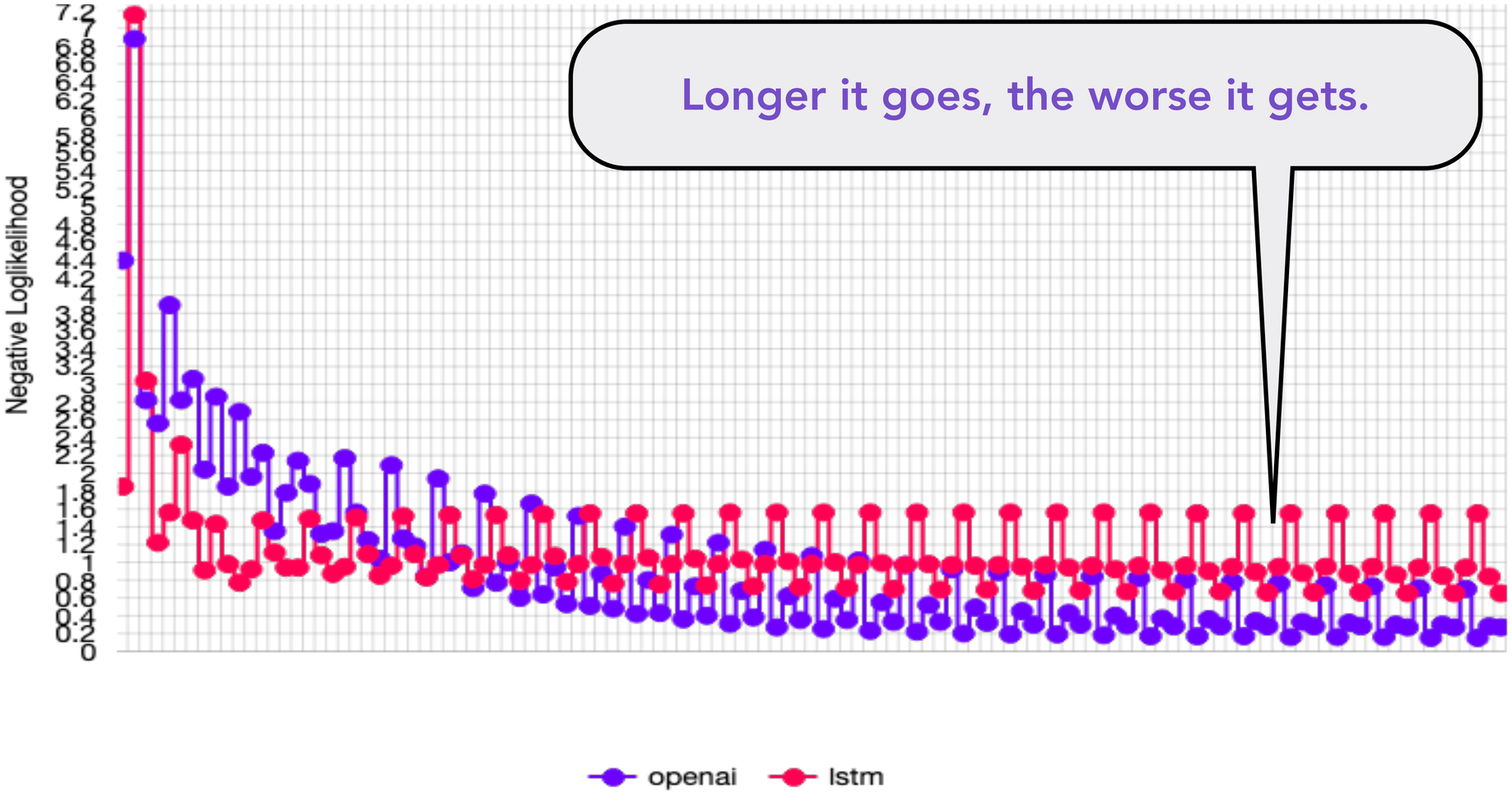
I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired.



Holtzman et. al., ICLR 2020

Beam search gets repetitive and repetitive

I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired.



Holtzman et. al., ICLR 2020

How can we reduce repetition?

- Don’t repeat *n*-grams (Hacky, but works!)

- Minimize additional loss term for minimizing hidden state similarity (LSTMs)

$$\hat{y}_t = g\Big(\log P(y_t \mid \{y\}_{<t}) - s(h_t, h_{t-m})\Big)$$

- Select token that maximizes mutual information between target and history

$$\hat{y}_t = g\Big(\log P(y_t \mid \{y\}_{<t}) - \log P(y_t)\Big)$$

(Li et. al., NAACL 2016, Celikyilmaz et al., NAACL 2018, Paulus et al., NAACL 2018)

How can we reduce repetition?

- Don't repeat *n*-grams (Hacky, but v

Minimize **hidden state similarity** in LSTMs

- Minimize additional loss term for minimizing hidden state similarity (LSTMs)

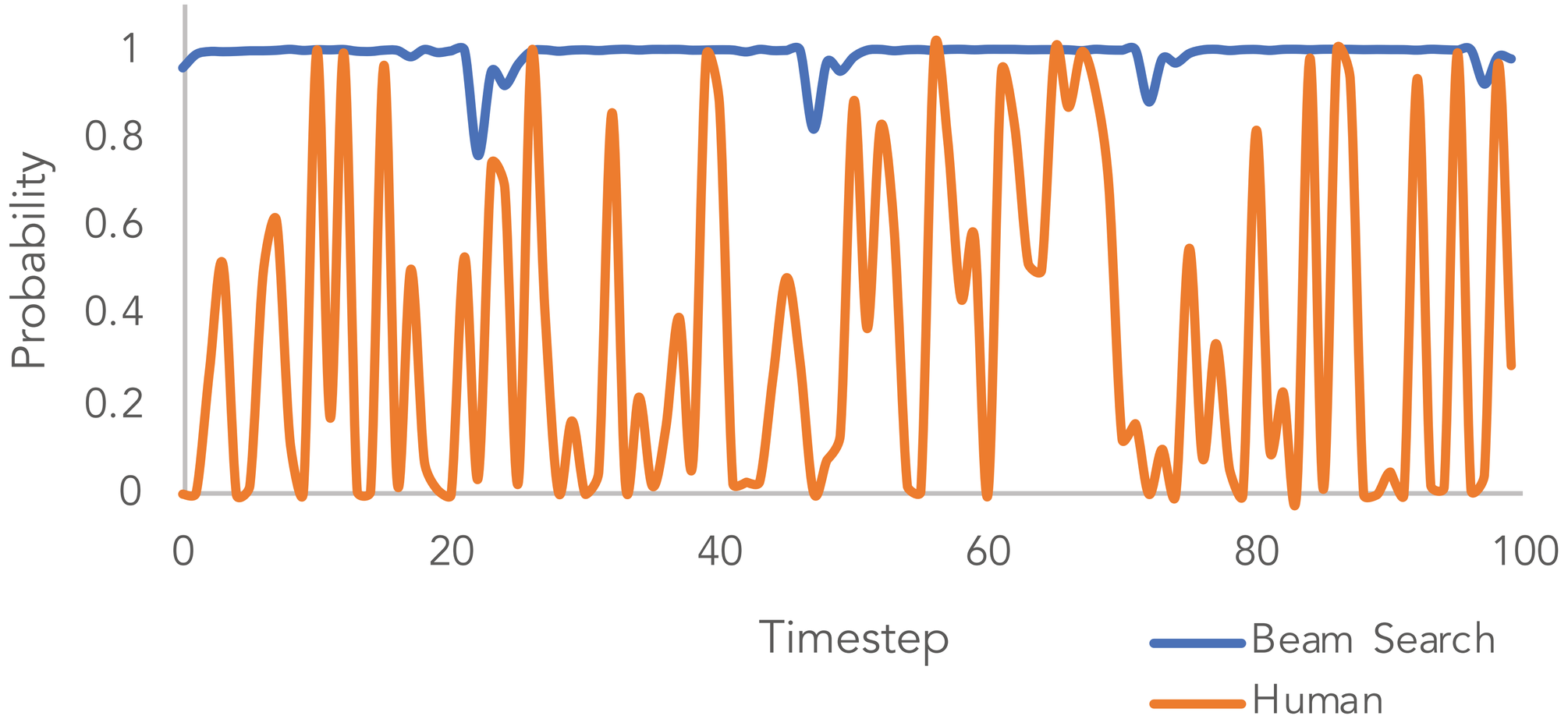
$$\hat{y}_t = g\Big(\log P(y_t \mid \{y\}_{<t}) - s(h_t, h_{t-m})\Big)$$

- Select token that maximizes mutual information between target and history

$$\hat{y}_t = g\Big(\log P(y_t \mid \{y\}_{<t}) - \log P(y_t)\Big)$$

(Li et. al., NAACL 2016, Celikyilmaz et al., NAACL 2018, Paulus et al., NAACL 2018)

Step-by-step Maximization



Holtzman et. al., ICLR 2020

Time to get *random*: Sampling

- g = sample a token from the distribution of tokens

$$\hat{y}_t \sim P(y_t = w \mid \{y\}_{<t})$$

Whoa, too *random*: Temperature Scaling

- Too much randomness: distribution has too much entropy

Whoa, too *random*: Temperature Scaling

- Too much randomness: distribution has too much entropy
- **Solution:** Make the distribution more "peaky" with temperature scaling

Recall: $P(y_t \mid \{y\}_{<t}) = \frac{e^{o_n}}{\sum_{m=1}^{M} e^{o_m}}$

Whoa, too *random*: Temperature Scaling

- Too much randomness: distribution has too much entropy
- **Solution:** Make the distribution more "peaky" with temperature scaling

Recall: $P(y_t \mid \{y\}_{<t}) = \frac{e^{o_n}}{\sum_{m=1}^{M} e^{o_m}}$

$$\Rightarrow \quad P(y_t \mid \{y\}_{<t}) = \frac{e^{o_n / \tau}}{\sum_{m=1}^{M} e^{o_m / \tau}}$$

Whoa, too *random*: Temperature Scaling

- Too much randomness: distribution has too much entropy
- **Solution:** Make the distribution more "peaky" with temperature scaling

$$P(y_t \mid \{y\}_{<t}) = \frac{e^{o_n / \tau}}{\sum_{m=1}^{M} e^{o_m / \tau}}$$

$\tau > 1$ "flatter" distribution

$\tau < 1$ "peakier" distribution

Maybe we need fewer options: Top-*k* sampling

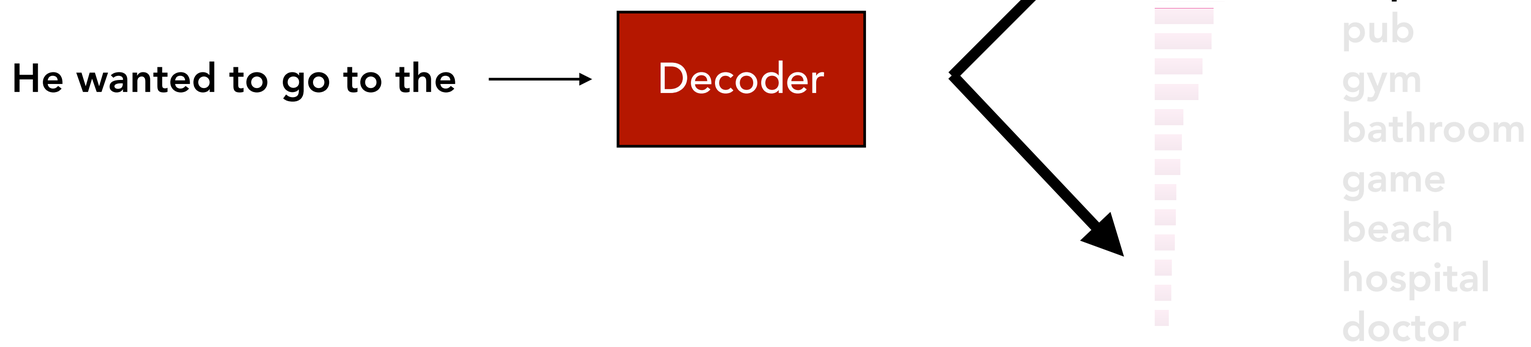
- The entire distribution over tokens is not needed at every step
- Many token choices should have no chance of being selected

(Holtzman et. al., ACL 2018; Fan et al., ACL 2018)

Maybe we need fewer options: Top-*k* sampling

- The entire distribution over tokens is not needed at every step
- Many token choices should have no chance of being selected
- Only sample from the top *k* tokens in the distribution

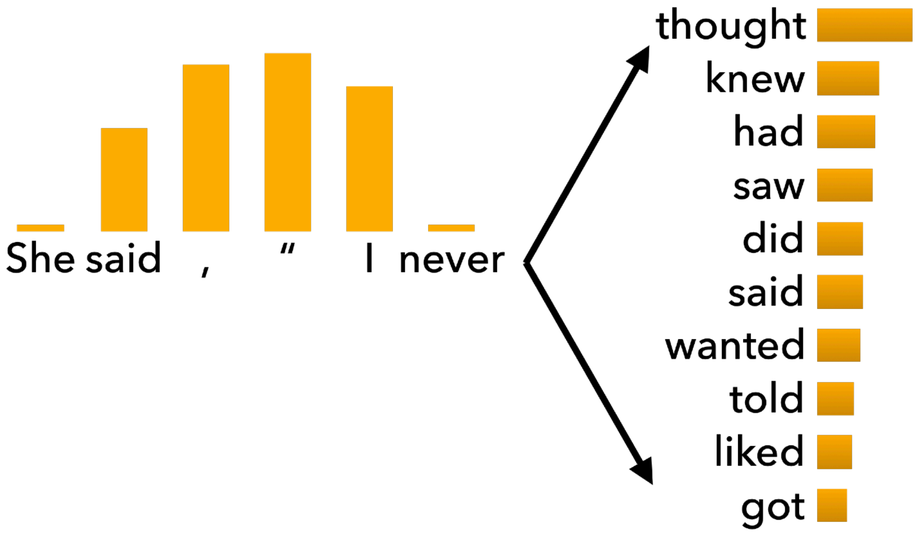
$$\hat{y}_t \sim P^*(y_t = w \mid \{y\}_{<t})$$



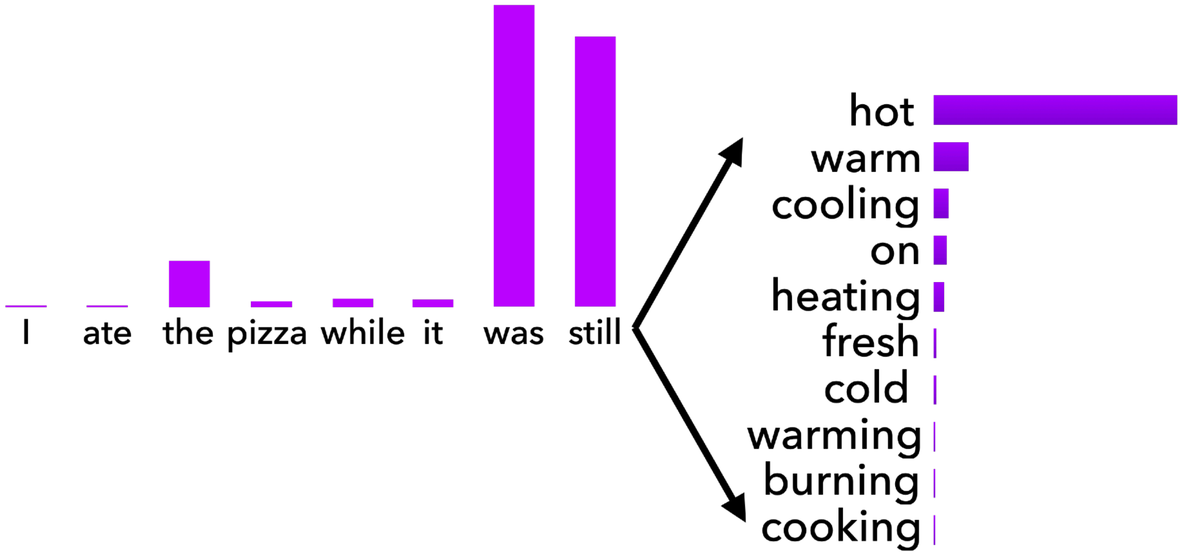
(Holtzman et. al., ACL 2018; Fan et al., ACL 2018)

Issues with top-*k* sampling

Top-*k* can cut-off too ***quickly***



Top-*k* can cut-off too ***slowly***

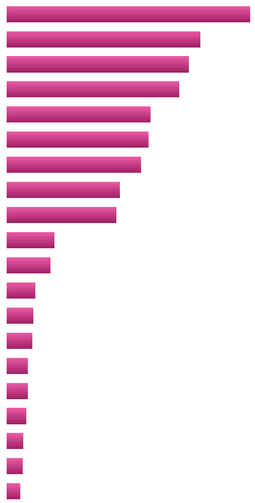


(Holtzman et. al., ICLR 2020)

I don’t know how many options I need: Top-*p* sampling

- Also known as **nucleus** sampling
- Sample from subset of vocabulary where probability mass is concentrated

$P_1(y_t \mid \{y\}_{<t})$



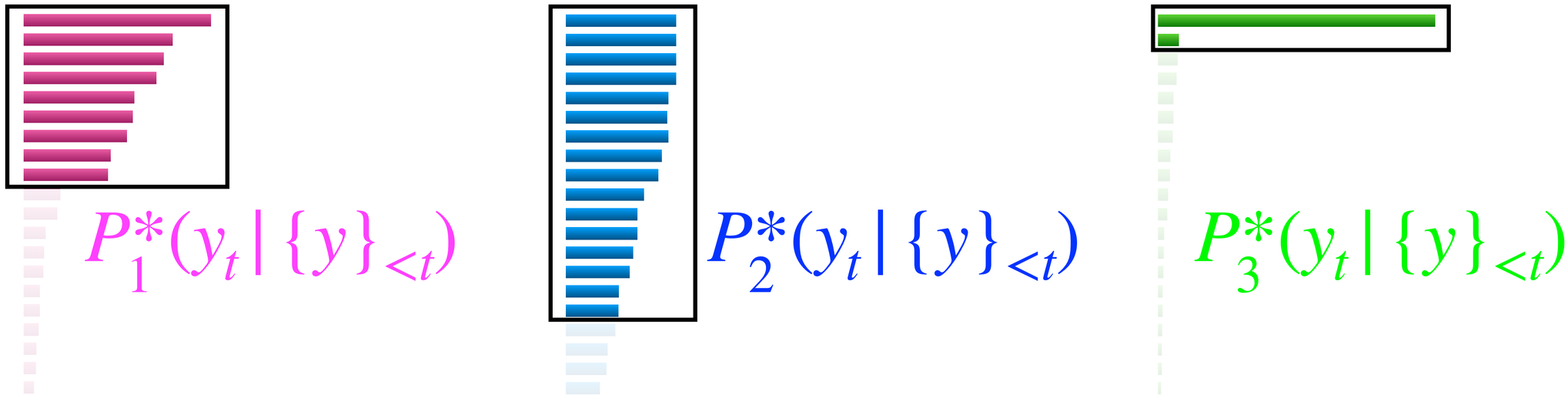
$P_2(y_t \mid \{y\}_{<t})$

$P_3(y_t \mid \{y\}_{<t})$

(Holtzman et. al., ICLR 2020)

I don’t know how many options I need: Top-*p* sampling

- Also known as **nucleus** sampling:
- Sample from subset of vocabulary where probability mass is concentrated
- Probability mass has a dynamically changing **nucleus**



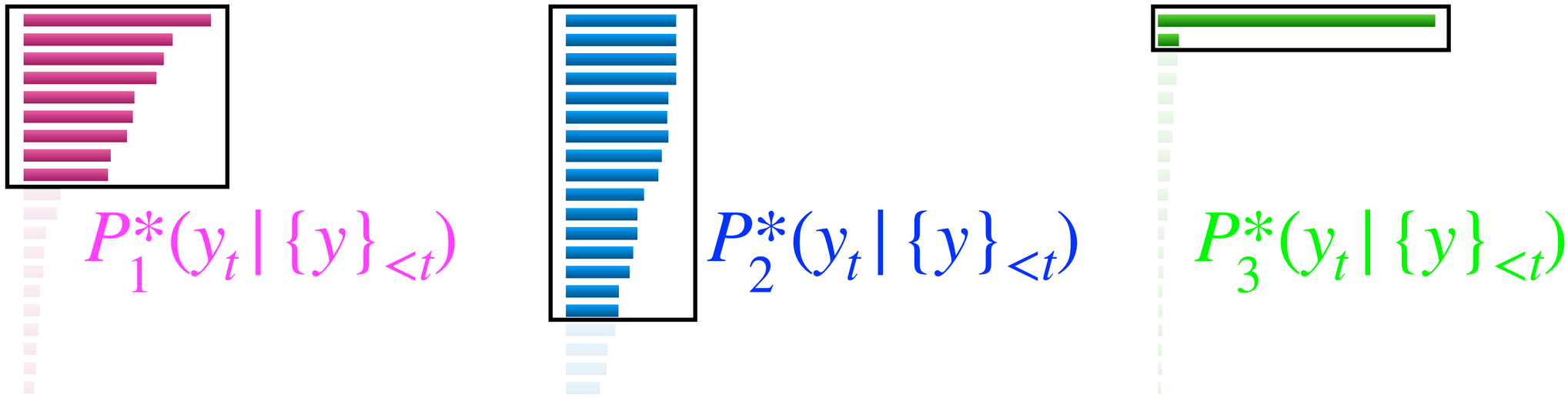
*nuclei not to scale

(Holtzman et. al., ICLR 2020)

I don’t know how many options I need: Top-*p* sampling

- Also known as **nucleus** sampling:
- Sample from subset of vocabulary where probability mass is concentrated
- Probability mass has a dynamically changing **nucleus**

Only sample from the **nucleus** of the **distribution**



*nuclei not to scale

(Holtzman et. al., ICLR 2020)

This all sounds a bit *risky*

- What if my distribution isn’t great to begin with? Can I mitigate this?

This all sounds a bit *risky*

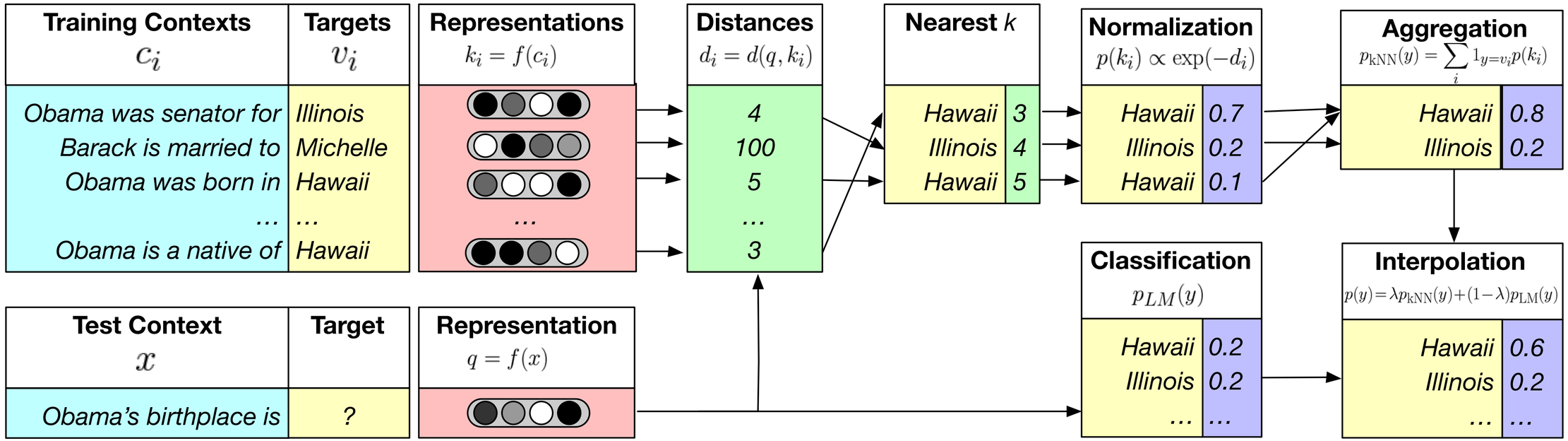
- What if my distribution isn't great to begin with? Can I mitigate this?

1. Make your word-level distributions more robust

kNN Language Models

- Don’t just rely on your trained model to generate a distribution over tokens

- Use knowledge of similar contexts from another corpus



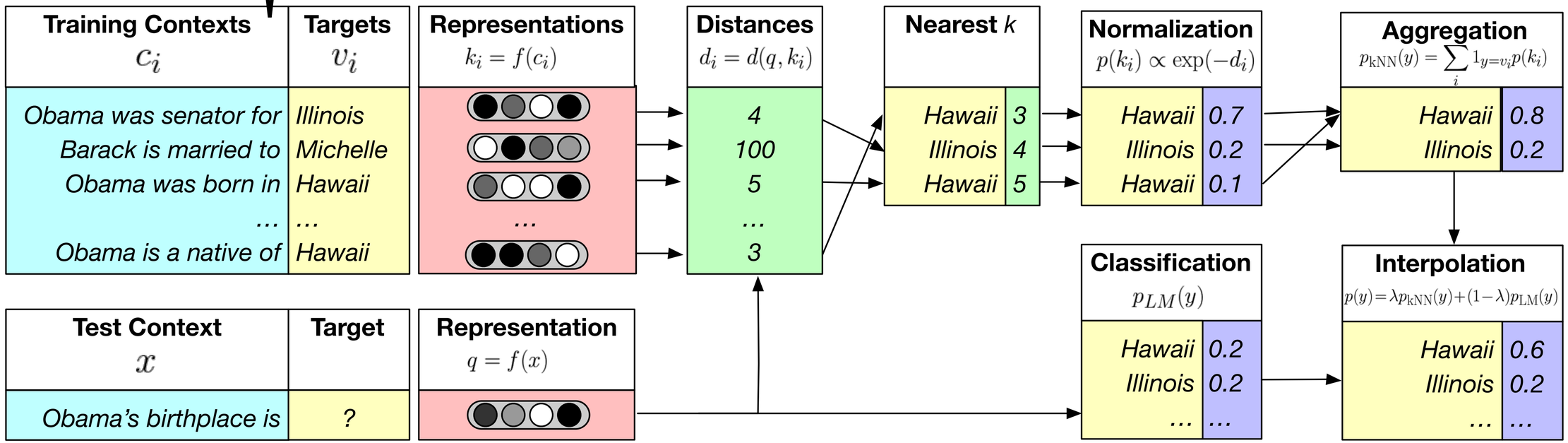
(Khandelwal et. al., ICLR 2020)

kNN Language Models

- Don't just rely on your trained model to generate a distribution over

Initialize a **database** of contexts

- Use knowledge of similar contexts from another corpus

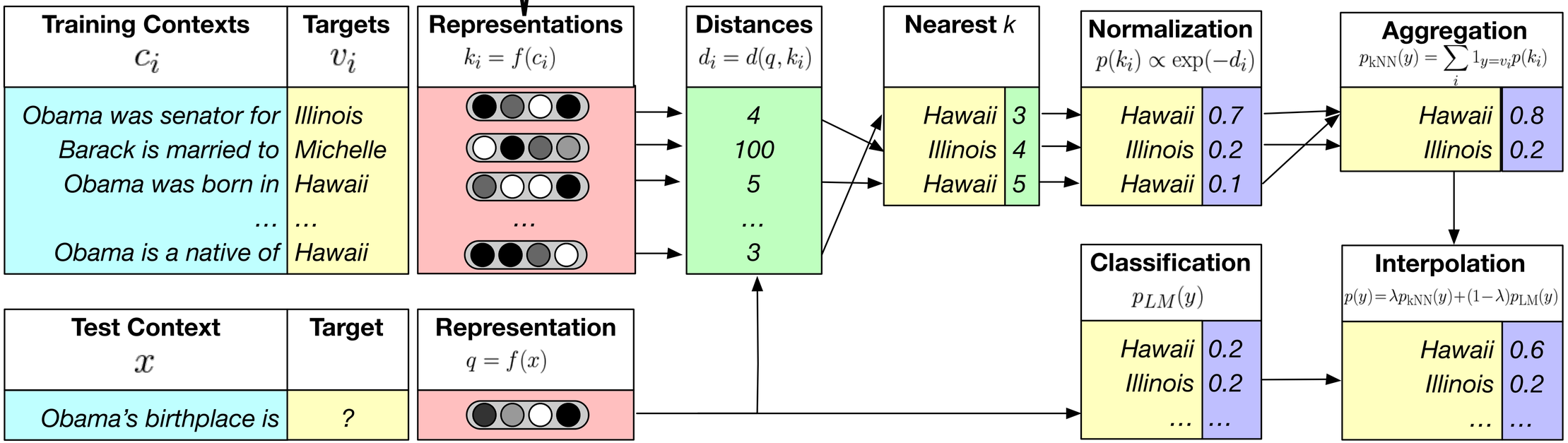


(Khandelwal et. al., ICLR 2020)

kNN Language Models

- Don't just rely on your trained model to generate a distribution over tokens
- Use knowledge of similar contexts from another corpus

Compute **representations** for each context in DB



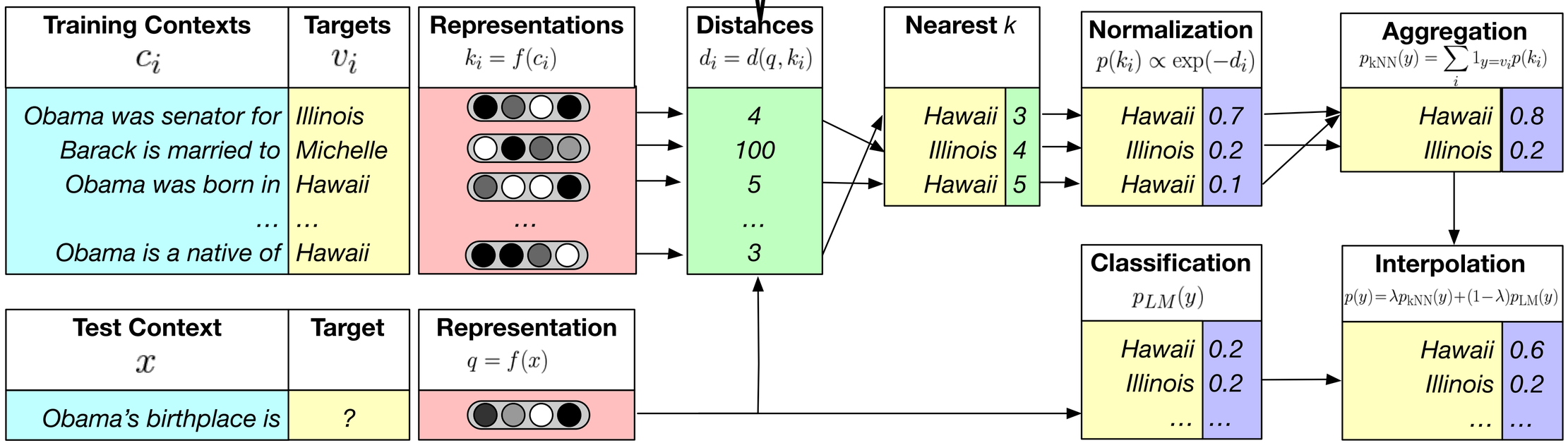
(Khandelwal et. al., ICLR 2020)

kNN Language Models

- Don't just rely on your trained model to generate a distribution over to

Efficiently compute **distance** between each context in DB and current sequence

- Use knowledge of similar contexts from another corpus

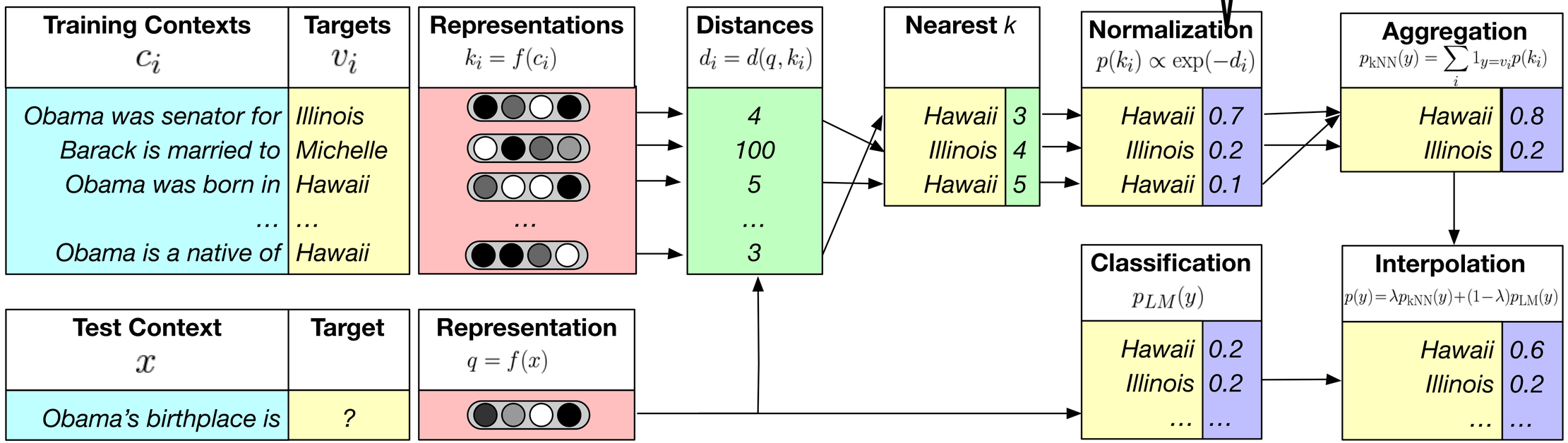


(Khandelwal et. al., ICLR 2020)

kNN Language Models

- Don't just rely on your trained model to generate a distribution over tokens
- Use knowledge of similar contexts from another corpus

Compute **distribution** over possible **targets** from context DB sequences using **distance of history**

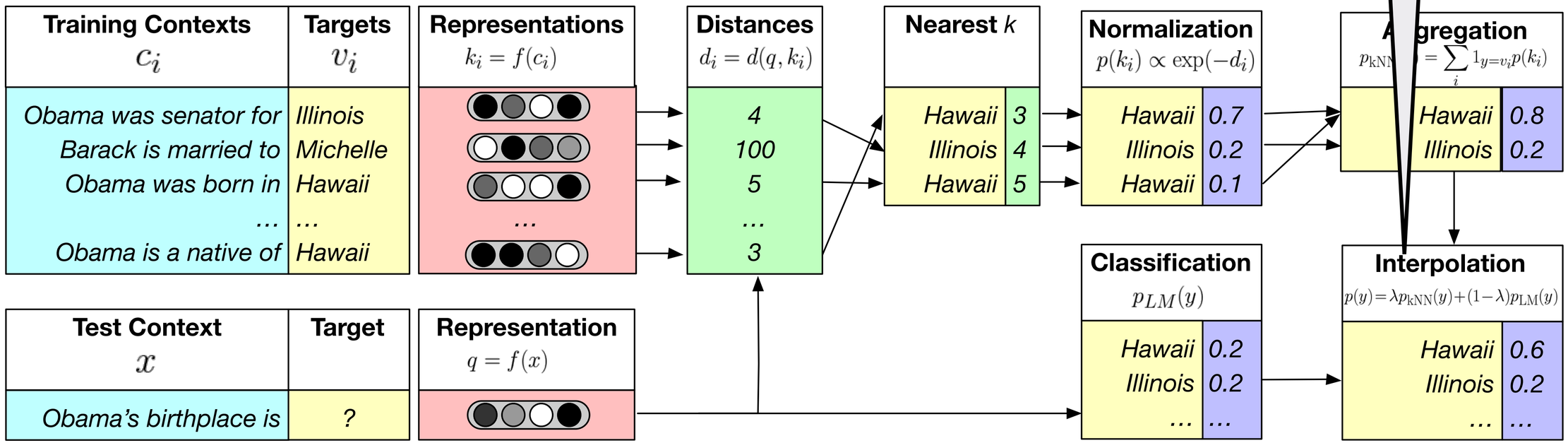


(Khandelwal et. al., ICLR 2020)

kNN Language Models

- Don't just rely on your trained model to generate a distribution over tokens
- Use knowledge of similar contexts from another corpus

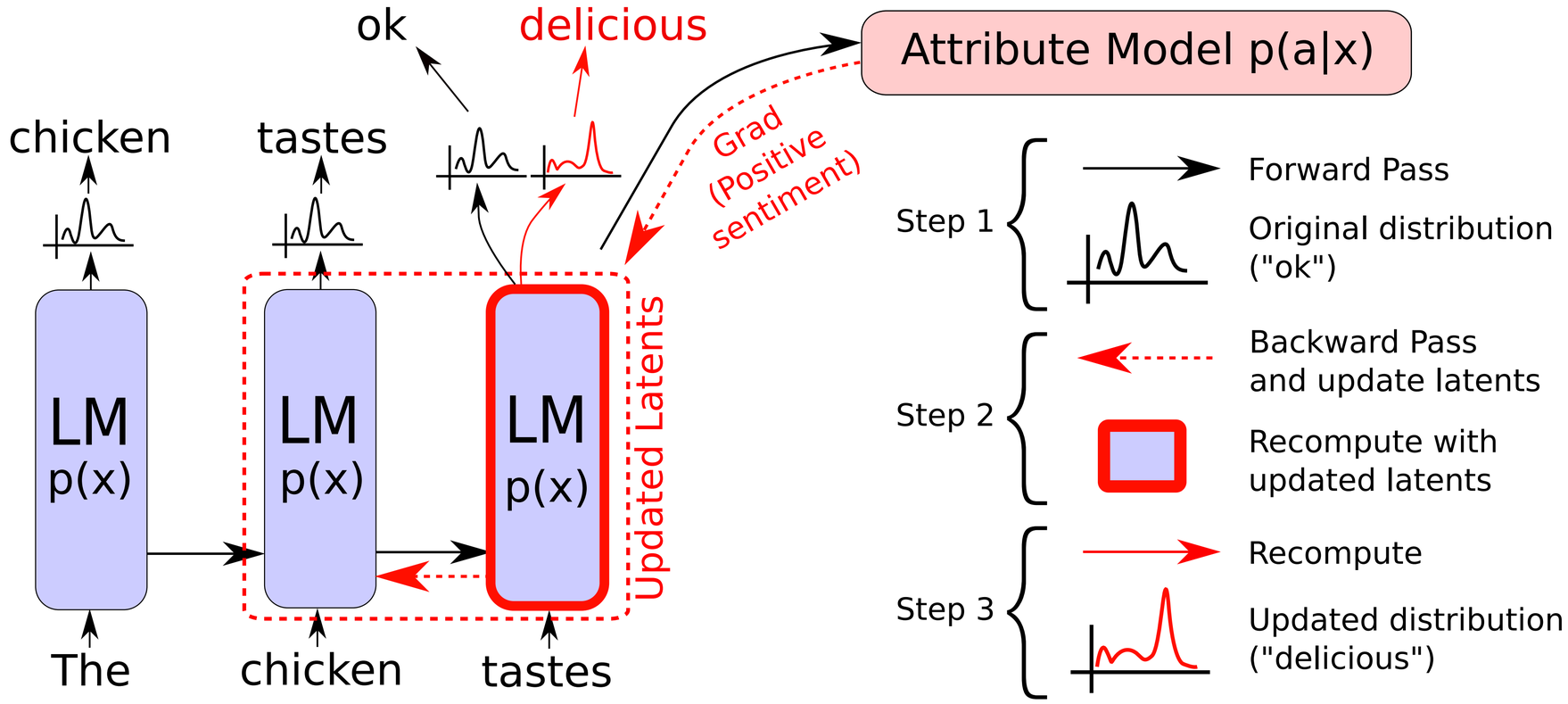
Interpolate distribution from nearest neighbor search with model distribution



(Khandelwal et. al., ICLR 2020)

Plug and Play Language Models!

- What if I want to encourage a tough to formalize behavior at inference time?

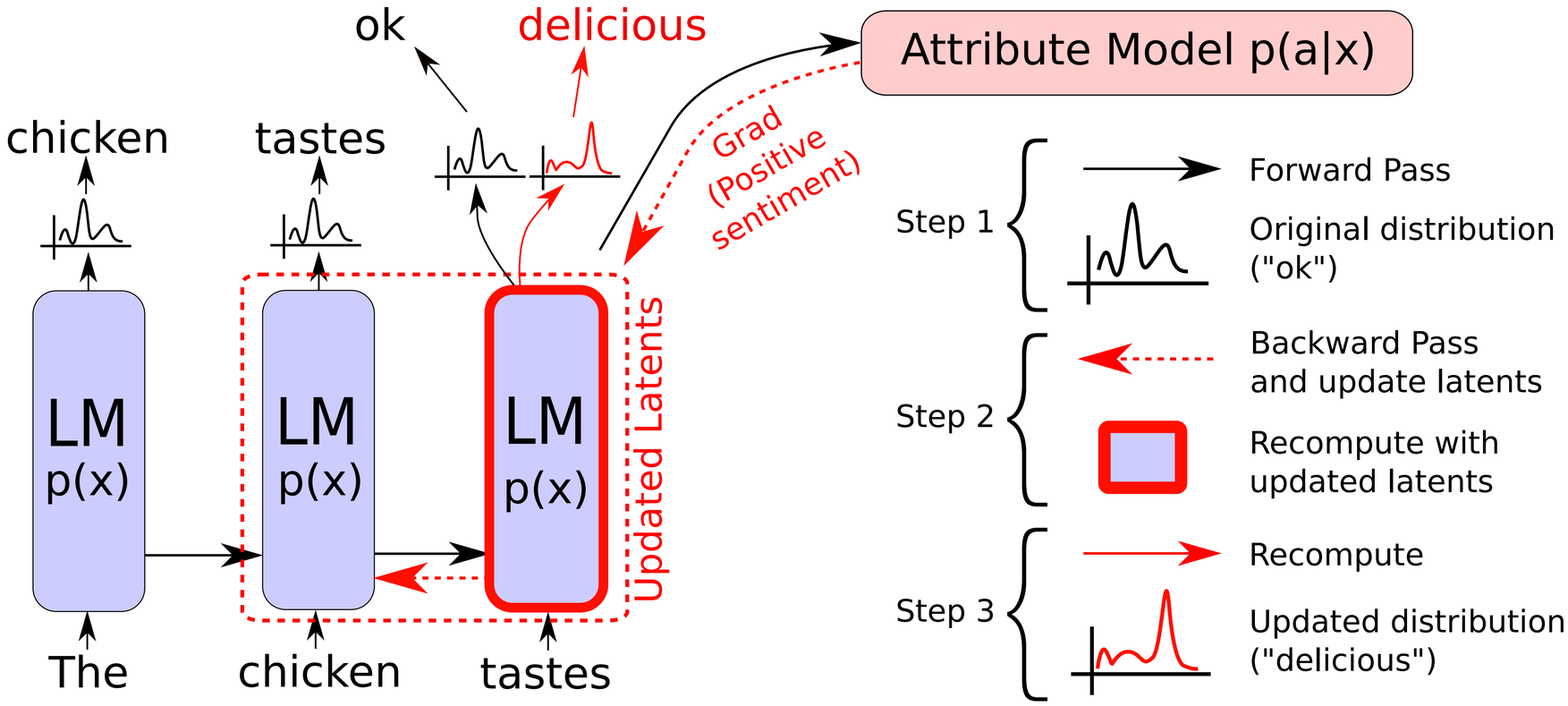


(Dathathri et al., ICLR 2020)

Plug and Pl s!

- What if I want to encou g time?

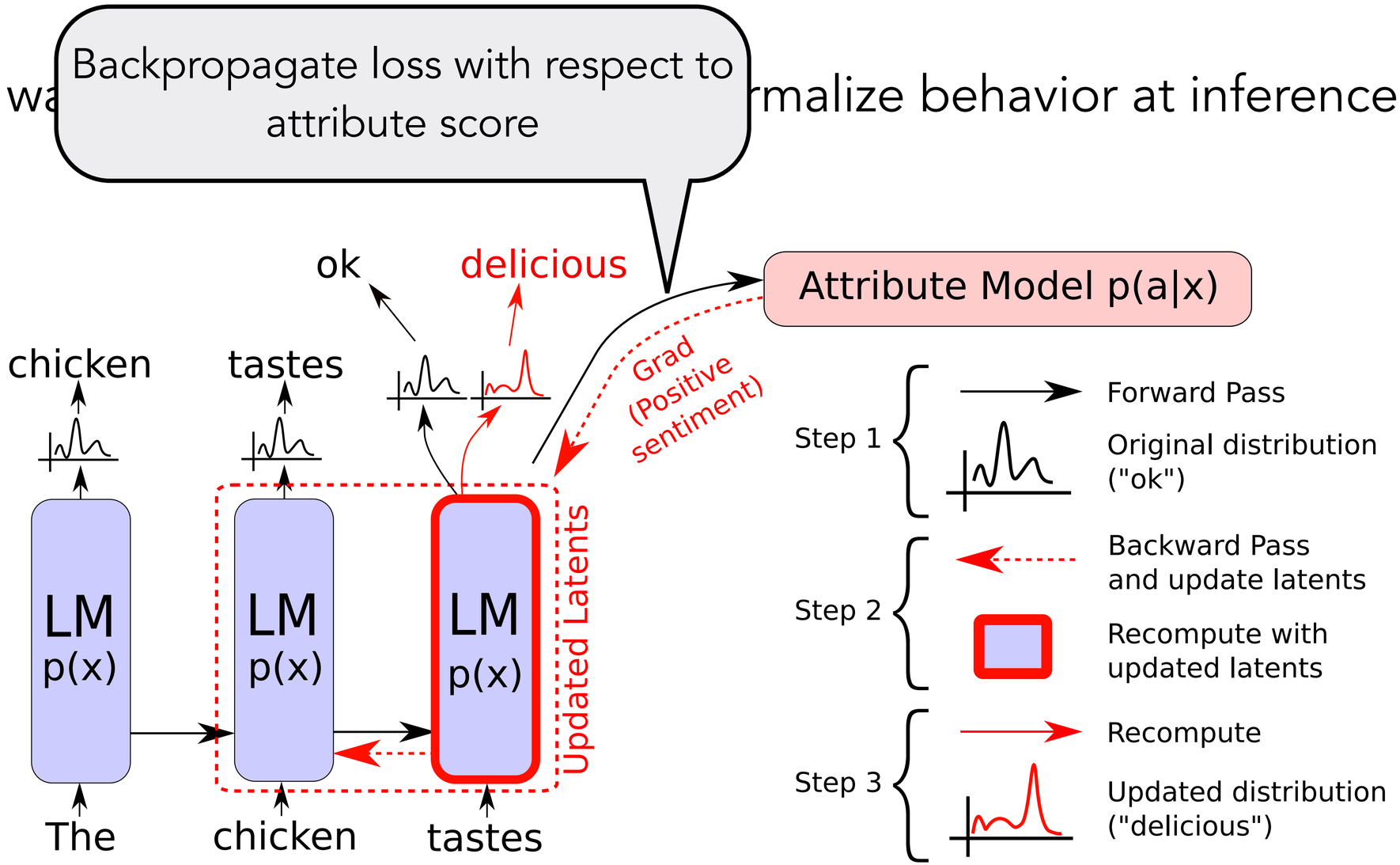
Define an attribute model that scores the generated sequence. Each generated token must try to increase the score given to the sequence by the attribute model



(Dathathri et al., ICLR 2020)

Plug and Play Language Models!

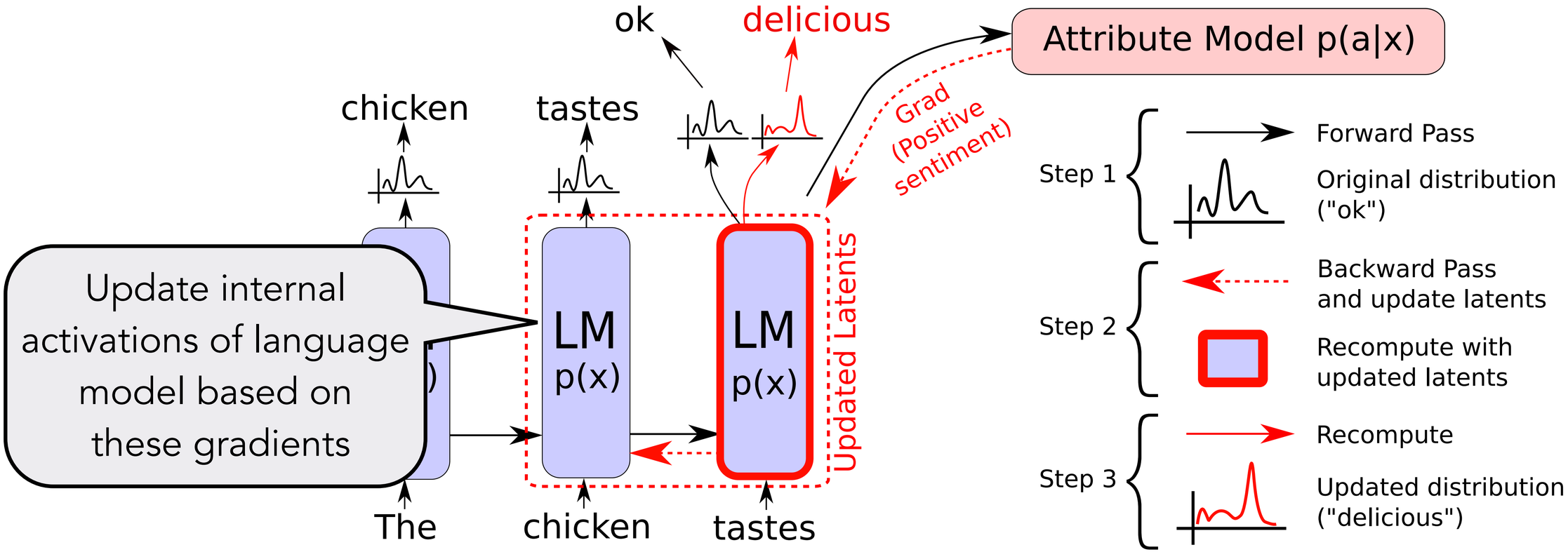
- What if I wa… rmalize behavior at inference time?



(Dathathri et al., ICLR 2020)

Plug and Play Language Models!

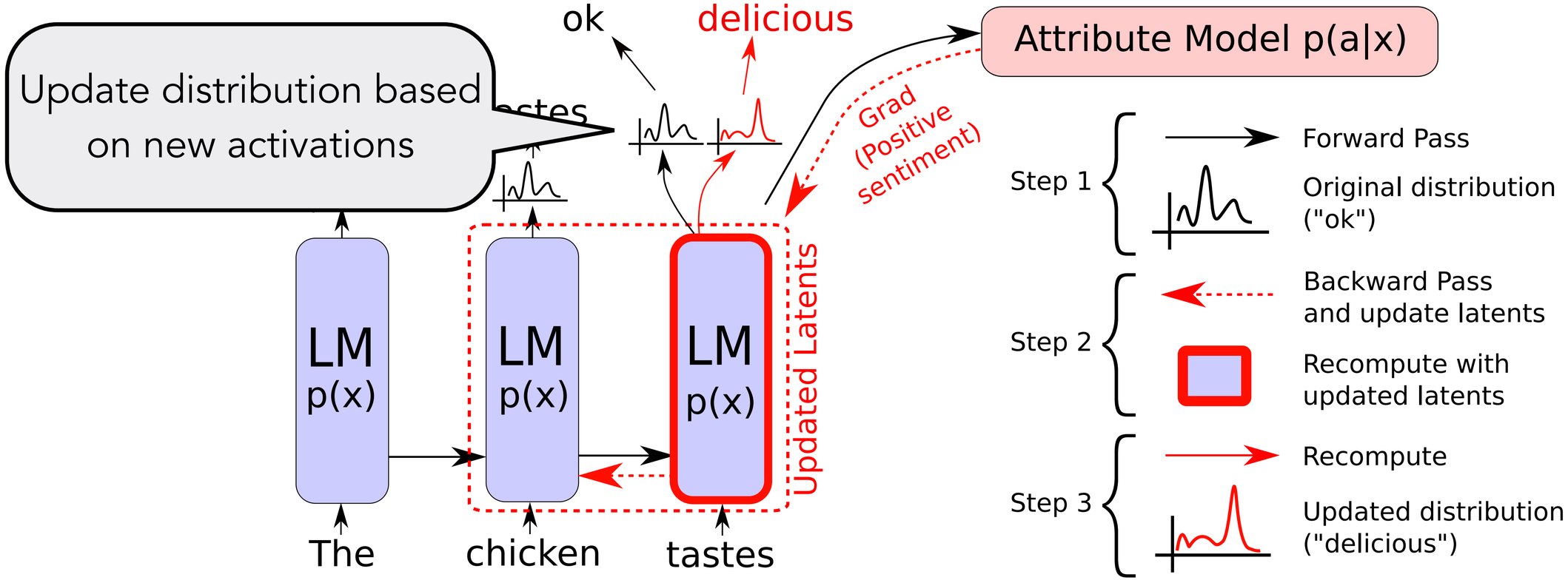
- What if I want to encourage a tough to formalize behavior at inference time?



(Dathathri et al., ICLR 2020)

Plug and Play Language Models!

- What if I want to encourage a tough to formalize behavior at inference time?



(Dathathri et al., ICLR 2020)

This all sounds a bit *risky*

- What if my distribution isn't great to begin with? Can I mitigate this?

1. Make your word-level distributions more robust
    - kNN LMs
    - Plug and play LMs

2. Optimize other sequence-level scores
    - Re-ranking
    - DeLorean

Re-ranking

- Generate a bunch of sequences with a sampling-based procedure
- Define a score to approximate the quality of your sequence.
- Simplest is to just use perplexity!
- However, re-rankers can be used to score a variety of properties: style (Holtzman et al., 2018), discourse (Gabriel et al., 2019), entailment (Pasunuru et al., 2018), logical consistency (Li et al., 2020; Lu et al., 2020)
- TODO: Make this more exciting

DELOREAN

(DEcoding for nonmonotonic LOgical REAsoNing)

(Qin et. al., EMNLP 2020)

Initialization

Any decoding algorithm will do!

Y

LM

x_1 x_2 ... x_{N_X}

Ray hung a tire on a rope to make his daughter a swing.

X

Ray ran to his daughter to make sure she was okay.

Z

Initialization

Any decoding algorithm will do!

Y

$\tilde{y}_1$ $\tilde{y}_2$ ... $\tilde{y}_N$

LM

x_1 x_2 ... x_{N_X}

Ray hung a tire on a rope to make his daughter a swing.

Ray ran to his daughter to make sure she was okay.

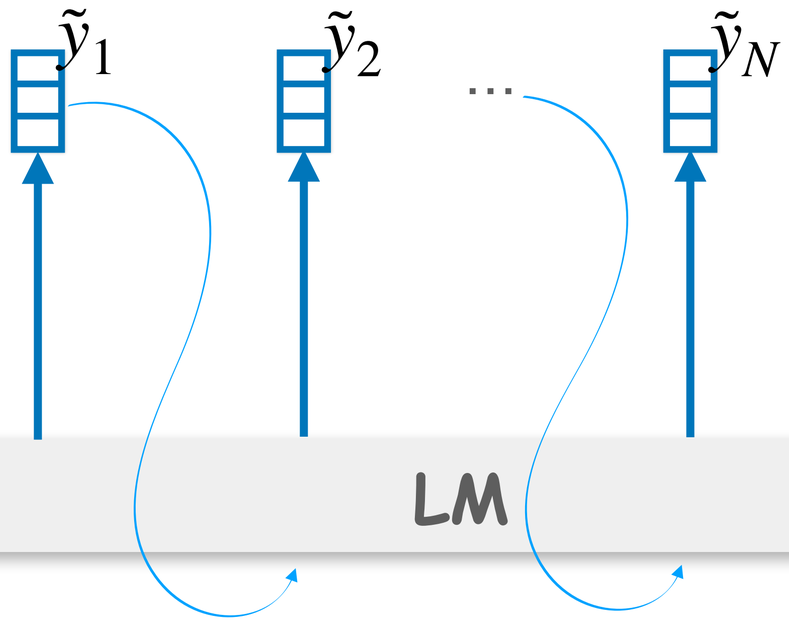
X

Z

Backward Pass

Backpropagate future information

Y



Ray hung a tire on a rope to make his daughter a swing.

X

Ray ran to his daughter to make sure she was okay.

Z

Backward Pass

Backpropagate future information

Y

$\tilde{y}_1$ $\tilde{y}_2$ ... $\tilde{y}_N$

Ray ran to ... [S]

LM

z_1 z_2 z_3 ... z_{N_Z}

Ray hung a tire on a rope to make his daughter a swing.

Ray ran to his daughter to make sure she was okay.

X

Z

Backward Pass

Backpropagate future information

Y

$\tilde{y}_1$ $\tilde{y}_2$ ... $\tilde{y}_N$

Ray ran to ... [S]

LM

z_1 z_2 z_3 ... z_{N_Z}

Ray hung a tire on a rope to make his daughter a swing.

Ray ran to his daughter to make sure she was okay.

X

Z

Backward Pass

Backpropagate future information

Y

$$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\text{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$$

$\tilde{y}_1$ $\tilde{y}_2$ ... $\tilde{y}_N$

Ray ran to ... [S]

LM

z_1 z_2 z_3 ... z_{N_Z}

Ray hung a tire on a rope to make his daughter a swing.

Ray ran to his daughter to make sure she was okay.

X

Z

Backward Pass

Backpropagate future information

Y

Backpropagation

$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\text{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$

$\tilde{y}_1$ $\tilde{y}_2$ ... $\tilde{y}_N$

Ray ran to ... [S]

LM

z_1 z_2 z_3 ... z_{N_Z}

Ray hung a tire on a rope to make his daughter a swing.

Ray ran to his daughter to make sure she was okay.

X

Z

Backward Pass

Backpropagate future information

Y

$\tilde{y}^b_1$ $\tilde{y}^b_2$ … $\tilde{y}^b_N$

Backpropagation

$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\text{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$

$\tilde{y}_1$ $\tilde{y}_2$ … $\tilde{y}_N$

Ray ran to … [S]

LM

z_1 z_2 z_3 … z_{N_Z}

Ray hung a tire on a rope to make his daughter a swing.

Ray ran to his daughter to make sure she was okay.

X

Z

Forward Pass

Mix both past and future information

Y

$\tilde{y}_1^b$ $\tilde{y}_2^b$ ... $\tilde{y}_N^b$

$\tilde{y}_1$ $\tilde{y}_2$ ... $\tilde{y}_N$

Backpropagation

$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\mathrm{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$

Ray ran to ... [S]

LM

z_1 z_2 z_3 ... z_{N_Z}

Ray hung a tire on a rope to make his daughter a swing.

X

Ray ran to his daughter to make sure she was okay.

Z

Forward Pass

Mix both past and future information

Y

$\tilde{y}_1^b$ $\tilde{y}_2^b$ ... $\tilde{y}_N^b$

$\tilde{y}_1$ $\tilde{y}_2$ ... $\tilde{y}_N$

Backpropagation

$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\mathrm{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$

Ray ran to ... [S]

LM

z_1 z_2 z_3 ... z_{N_Z}

Ray hung a tire on a rope to make his daughter a swing.

X

Ray ran to his daughter to make sure she was okay.

Z

Forward Pass

Mix both past and future information

Y

$\tilde{y}_1^b$ $\tilde{y}_2^b$ ... $\tilde{y}_N^b$

$\tilde{y}_1$ $\tilde{y}_2$ ... $\tilde{y}_N$

$\tilde{y}_1^f$ $\tilde{y}_2^f$ ... $\tilde{y}_N^f$

Backpropagation

$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\mathrm{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$

Ray ran to ... [S]

LM

x_1 x_2 ... x_{N_X}

z_1 z_2 z_3 ... z_{N_Z}

Ray hung a tire on a rope to make his daughter a swing.

Ray ran to his daughter to make sure she was okay.

X

Z

Iterate *T* times

Y

$\tilde{y}_1^b$ $\tilde{y}_2^b$ … $\tilde{y}_N^b$

$\tilde{y}_1$ $\tilde{y}_2$ … $\tilde{y}_N$

$\tilde{y}_1^f$ $\tilde{y}_2^f$ … $\tilde{y}_N^f$

Backpropagation

$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\mathrm{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$

Ray ran to … [S]

LM

x_1 x_2 … x_{N_X}

z_1 z_2 z_3 … z_{N_Z}

Ray hung a tire on a rope to make his daughter a swing.

Ray ran to his daughter to make sure she was okay.

X

Z

Sample final sequence

Y

$\tilde{y}^b_1$ $\tilde{y}^b_2$ … $\tilde{y}^b_N$

$\tilde{y}_1$ $\tilde{y}_2$ … $\tilde{y}_N$

$\tilde{y}^f_1$ $\tilde{y}^f_2$ … $\tilde{y}^f_N$

Backpropagation

$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\mathrm{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$

Ray ran to … [S]

LM

x_1 x_2 … x_{N_X}

z_1 z_2 z_3 … z_{N_Z}

Ray hung a tire on a rope to make his daughter a swing.

Ray ran to his daughter to make sure she was okay.

X

Z

Sample final sequence

Y

Output: She hit the rope and the tire fell on top of her.

Backpropagation

$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\mathrm{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$

$\tilde{y}^b_1$ $\tilde{y}^b_2$ … $\tilde{y}^b_N$

$\tilde{y}_1$ $\tilde{y}_2$ … $\tilde{y}_N$

$\tilde{y}^f_1$ $\tilde{y}^f_2$ … $\tilde{y}^f_N$

Ray ran to … [S]

LM

x_1 x_2 … x_{N_X}

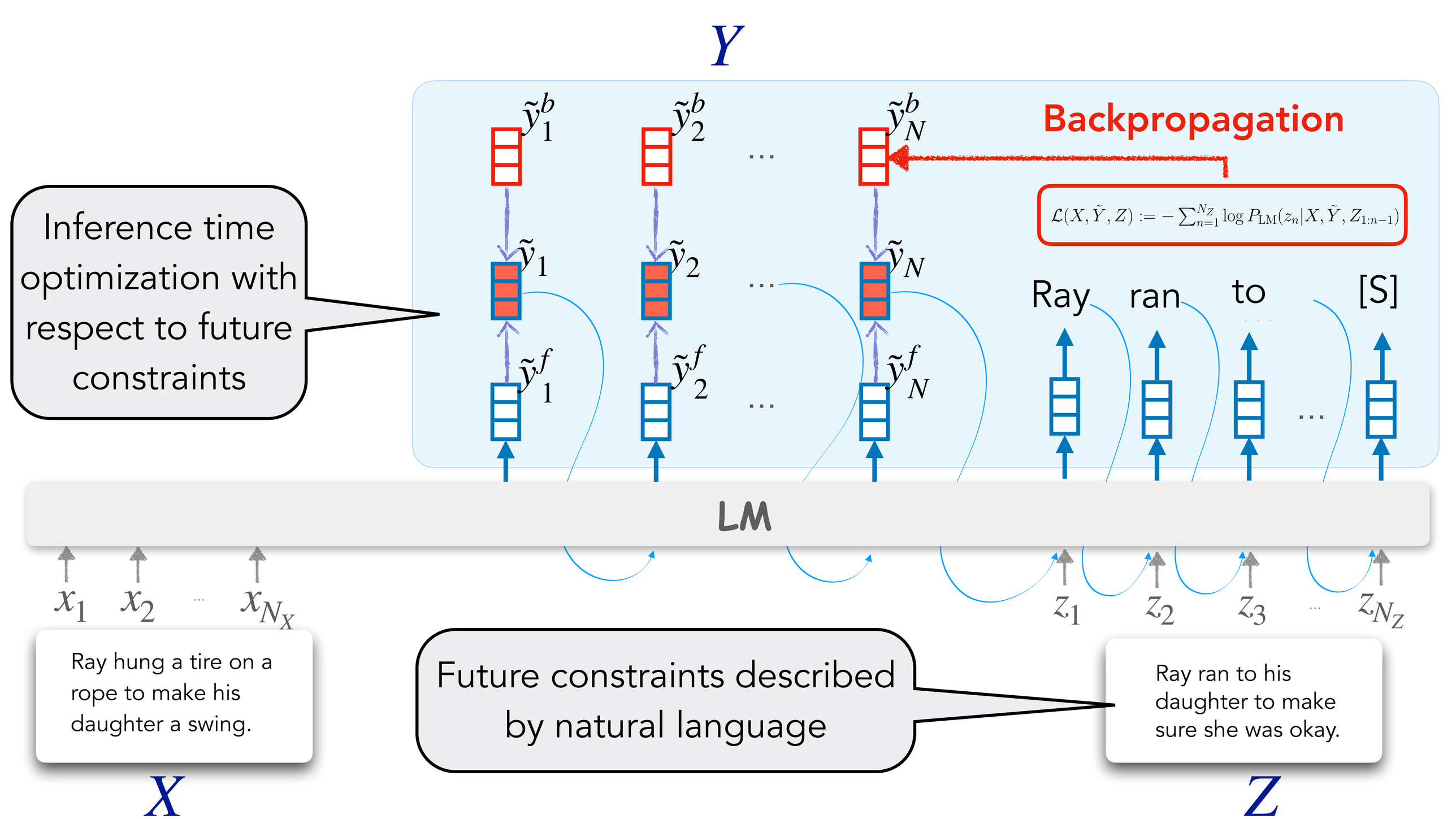
z_1 z_2 z_3 … z_{N_Z}

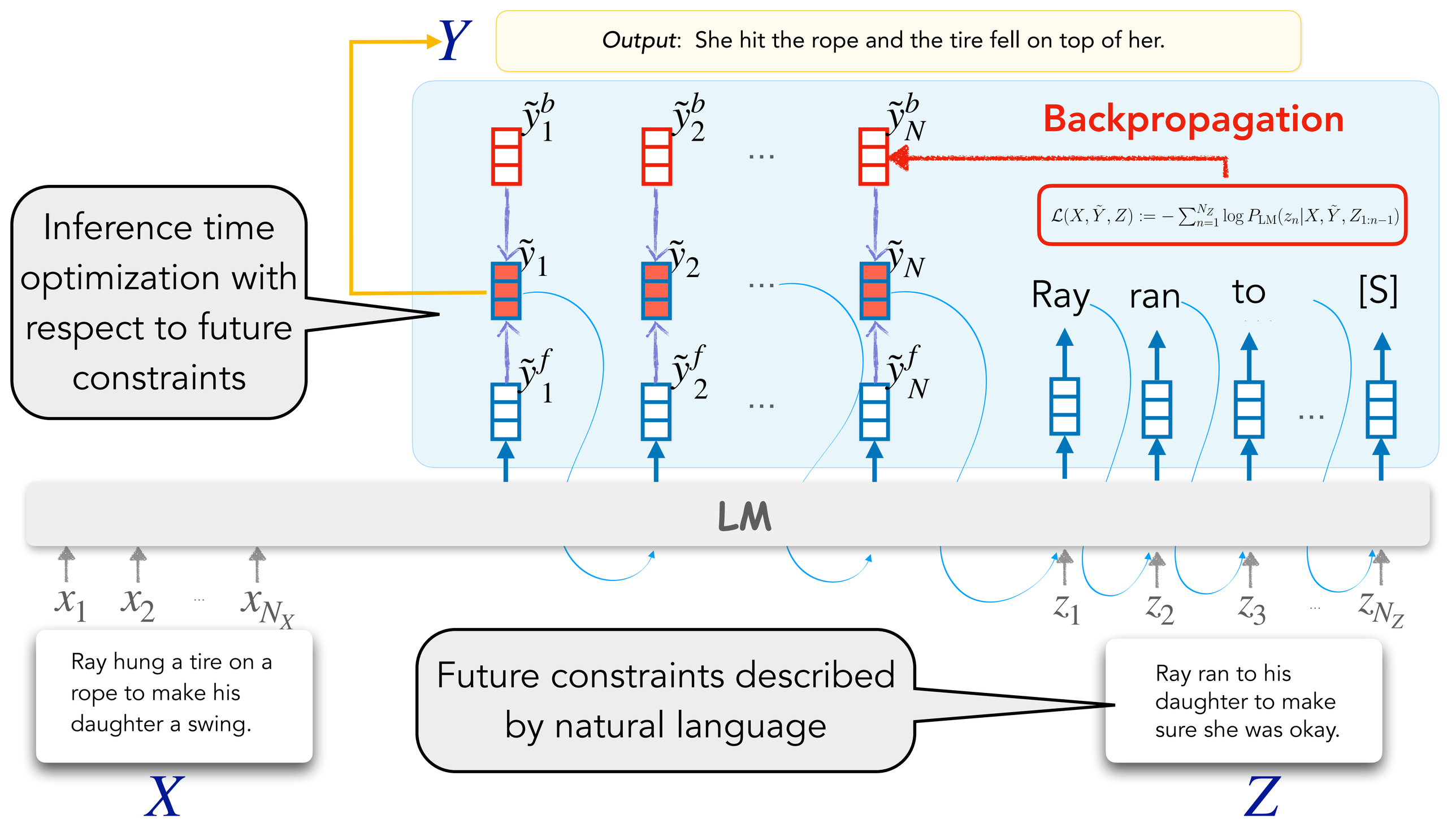
Ray hung a tire on a rope to make his daughter a swing.

Ray ran to his daughter to make sure she was okay.

X

Z


Y
Backpropagation
$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\mathrm{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$
Inference time optimization with respect to future constraints
Ray
ran
to
[S]
LM
x_1 x_2 ... x_{N_X}
z_1 z_2 z_3 ... z_{N_Z}
Ray hung a tire on a rope to make his daughter a swing.
Future constraints described by natural language
Ray ran to his daughter to make sure she was okay.
X
Z


Y
Output: She hit the rope and the tire fell on top of her.
Backpropagation
$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\mathrm{LM}}(z_n | X, \tilde{Y}, Z_{1:n-1})$
Inference time optimization with respect to future constraints
$\tilde{y}^b_1$ $\tilde{y}^b_2$ ... $\tilde{y}^b_N$
$\tilde{y}_1$ $\tilde{y}_2$ ... $\tilde{y}_N$
$\tilde{y}^f_1$ $\tilde{y}^f_2$ ... $\tilde{y}^f_N$
Ray ran to ... [S]
LM
x_1 x_2 ... x_{N_X}
z_1 z_2 z_3 ... z_{N_Z}
Ray hung a tire on a rope to make his daughter a swing.
Future constraints described by natural language
Ray ran to his daughter to make sure she was okay.
X
Z

Training Neural Text Generation Models

Antoine Bosselut

Generation Model Basics

1. At each time step, model computes a score o_n for each token in our vocabulary, $w_n \in V$

$$O_n = f(\{y\}_{<t})$$

$f(\,.\,)$ is your model

2. Compute a probability distribution over these scores (usually softmax)

$P(\,.\,)$ is your distribution over tokens

$$P(y_t = w_n \mid \{y\}_{<t}) = \frac{e^{o_n}}{\sum_{m=1}^{M} e^{o_m}}$$

Maximizing Likelihood

- Trained to generate the next word given a set of preceding words

Allen

y_0^*

Maximizing Likelihood

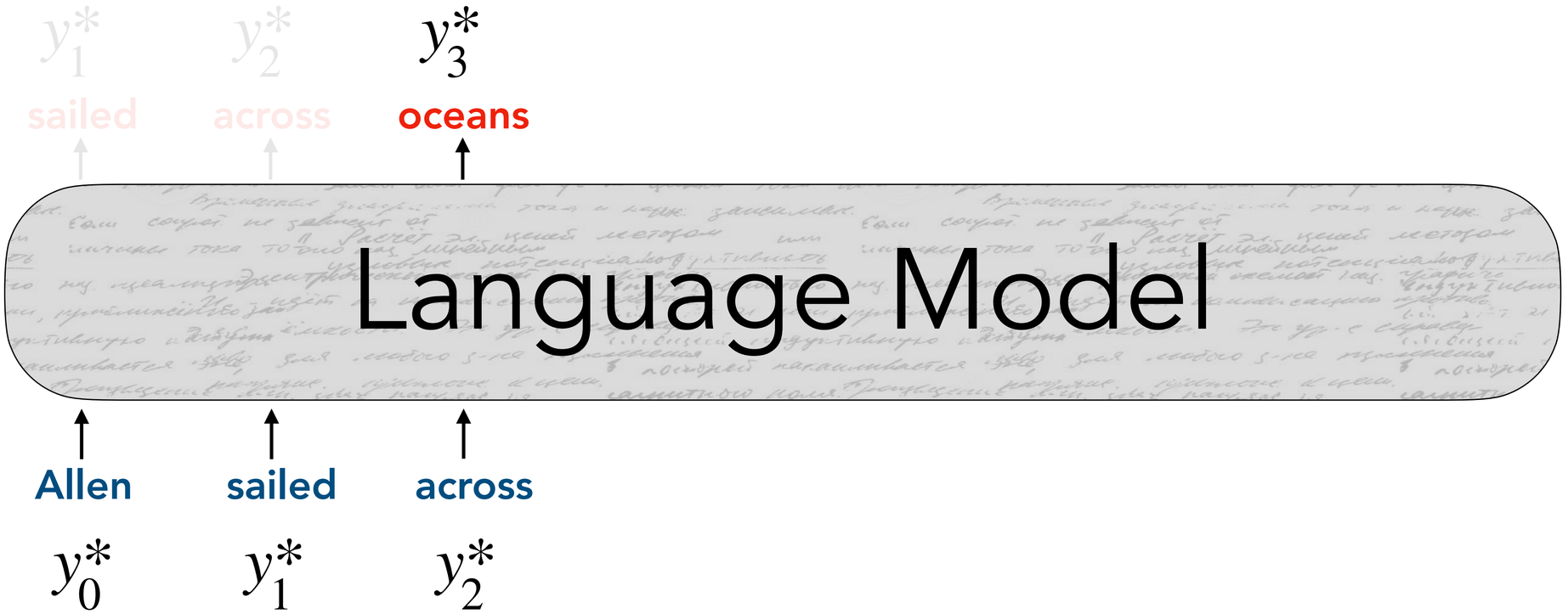
- Trained to generate the next word given a set of preceding words

Maximizing Likelihood

- Trained to generate the next word given a set of preceding words

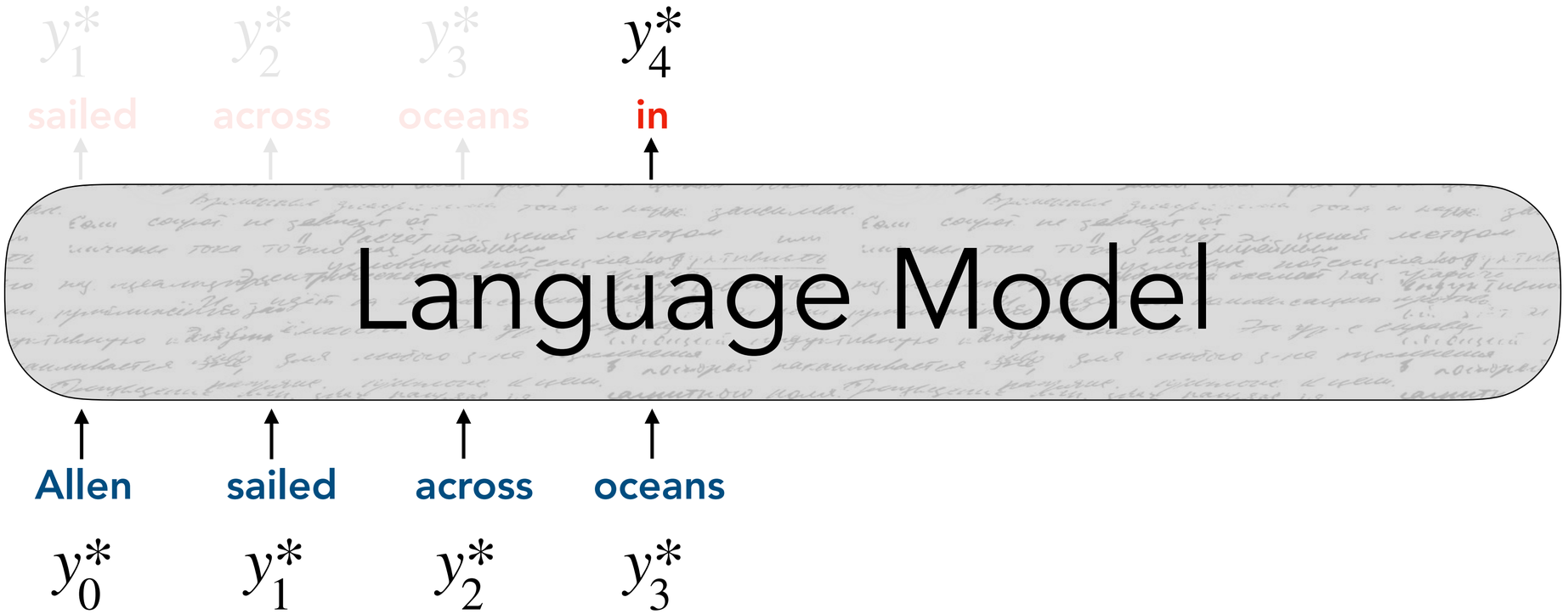
Maximizing Likelihood

- Trained to generate the next word given a set of preceding words



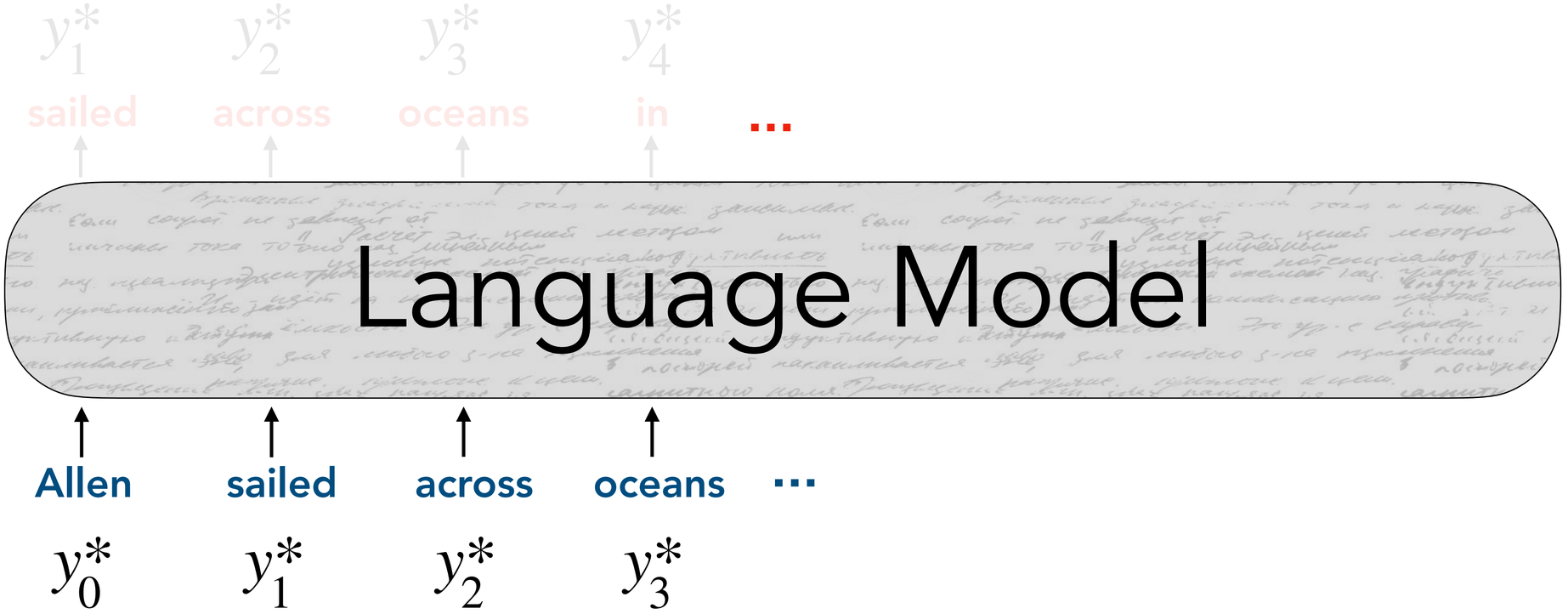
Maximizing Likelihood

- Trained to generate the next word given a set of preceding words



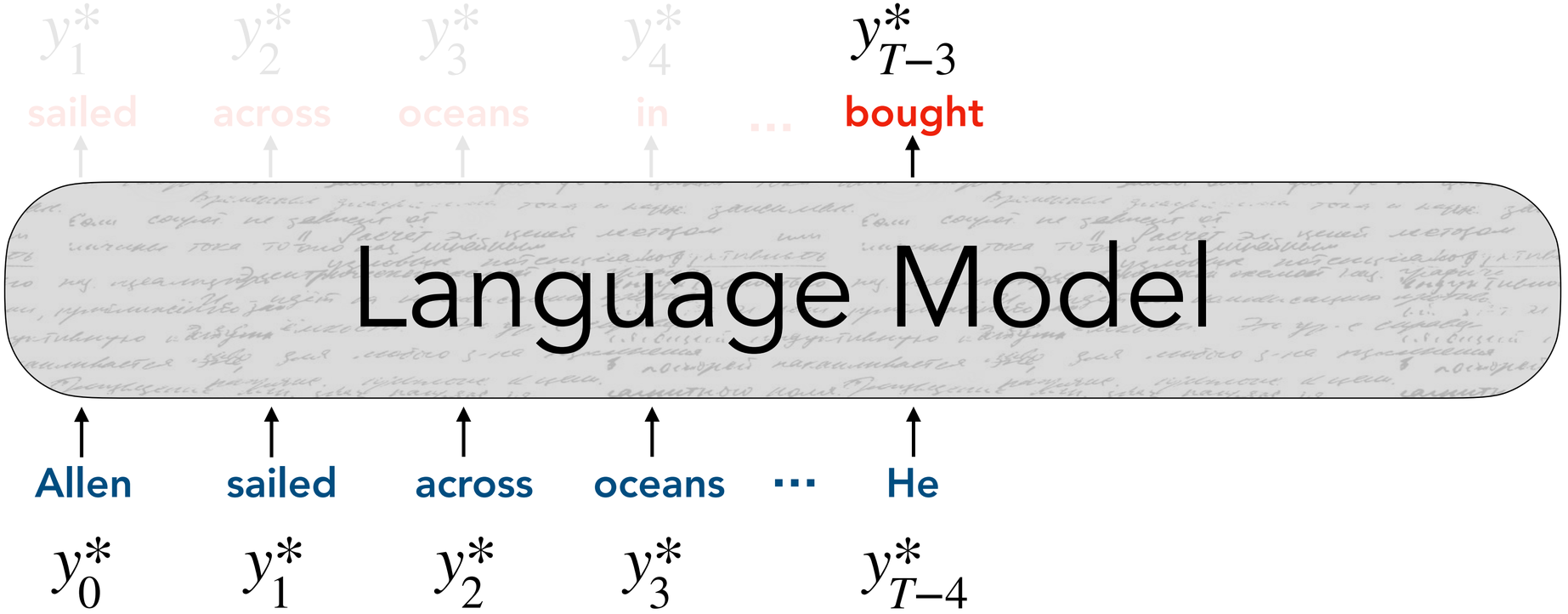
Maximizing Likelihood

- Trained to generate the next word given a set of preceding words



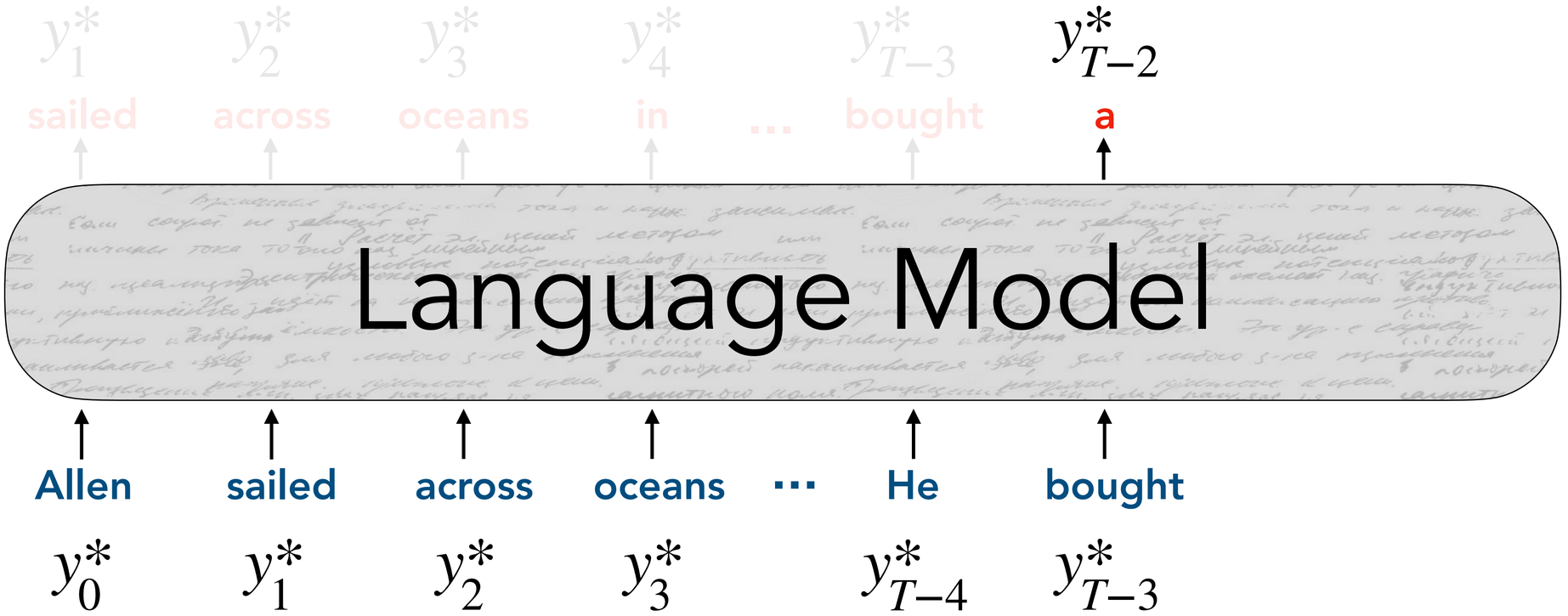
Maximizing Likelihood

- Trained to generate the next word given a set of preceding words



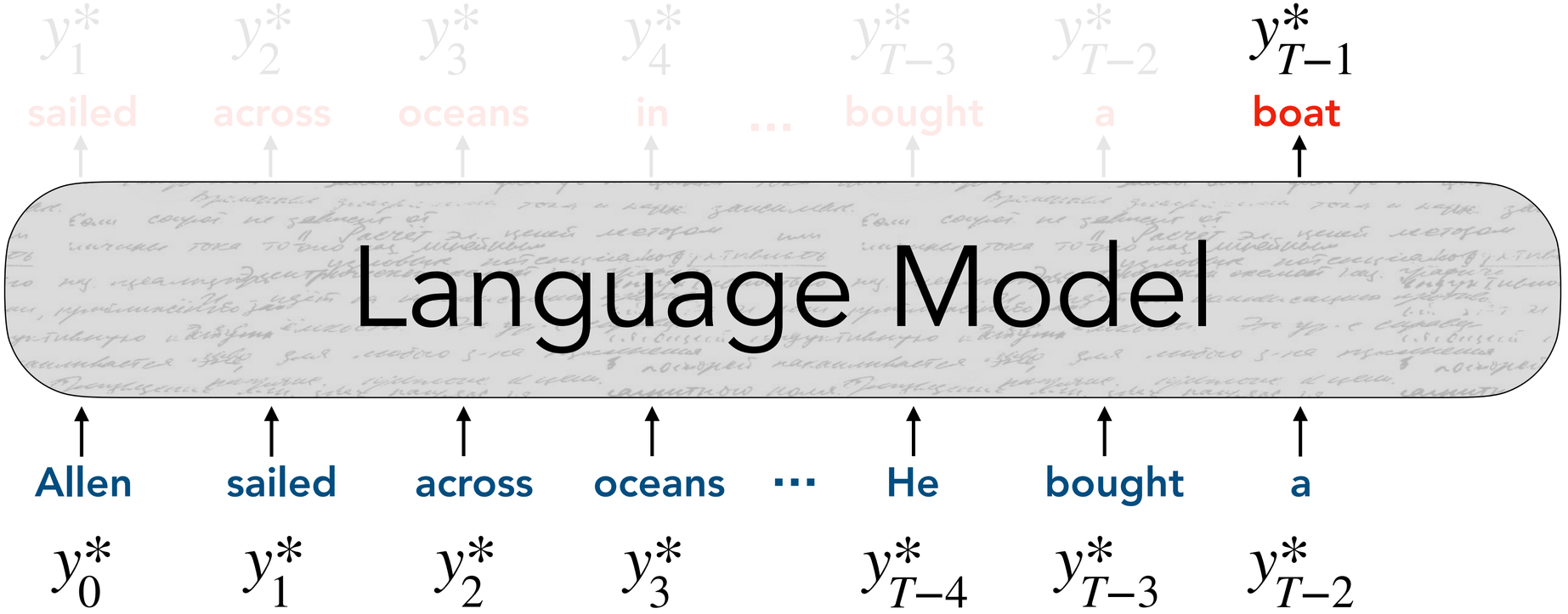
Maximizing Likelihood

- Trained to generate the next word given a set of preceding words



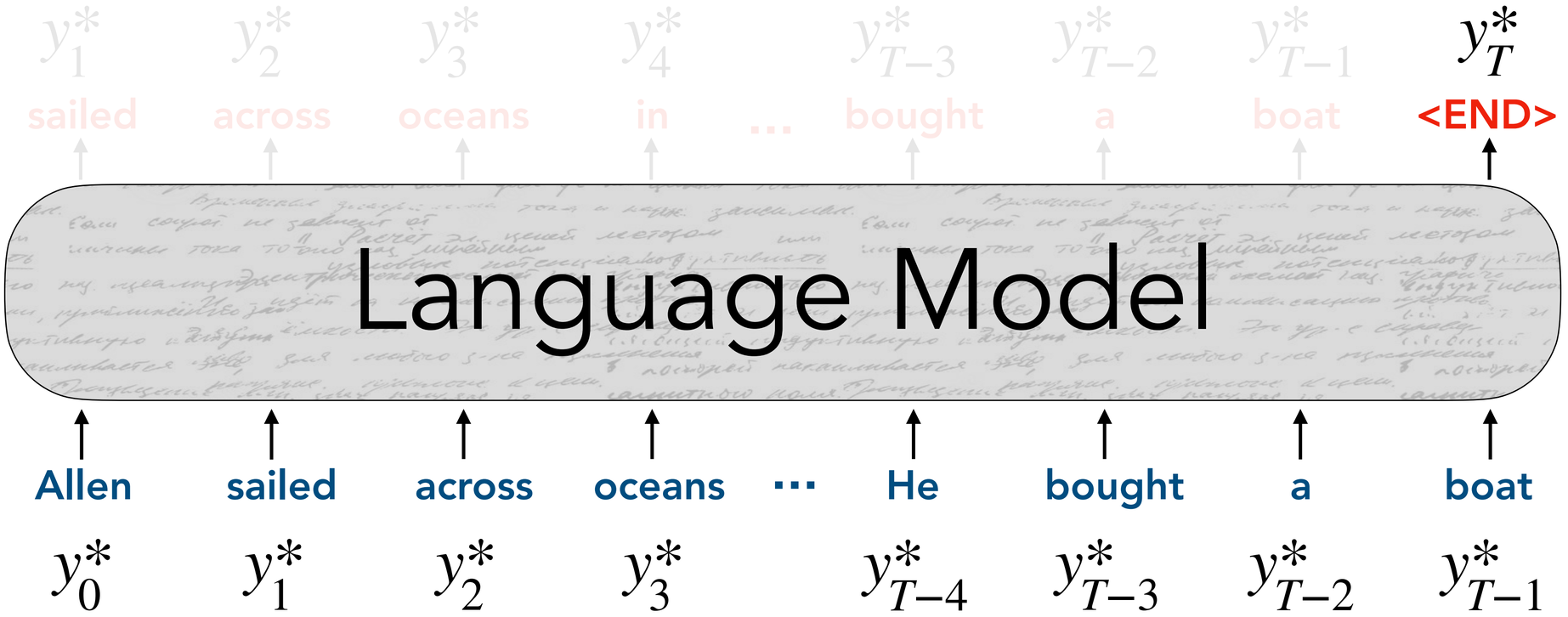
Maximizing Likelihood

- Trained to generate the next word given a set of preceding words



Maximizing Likelihood

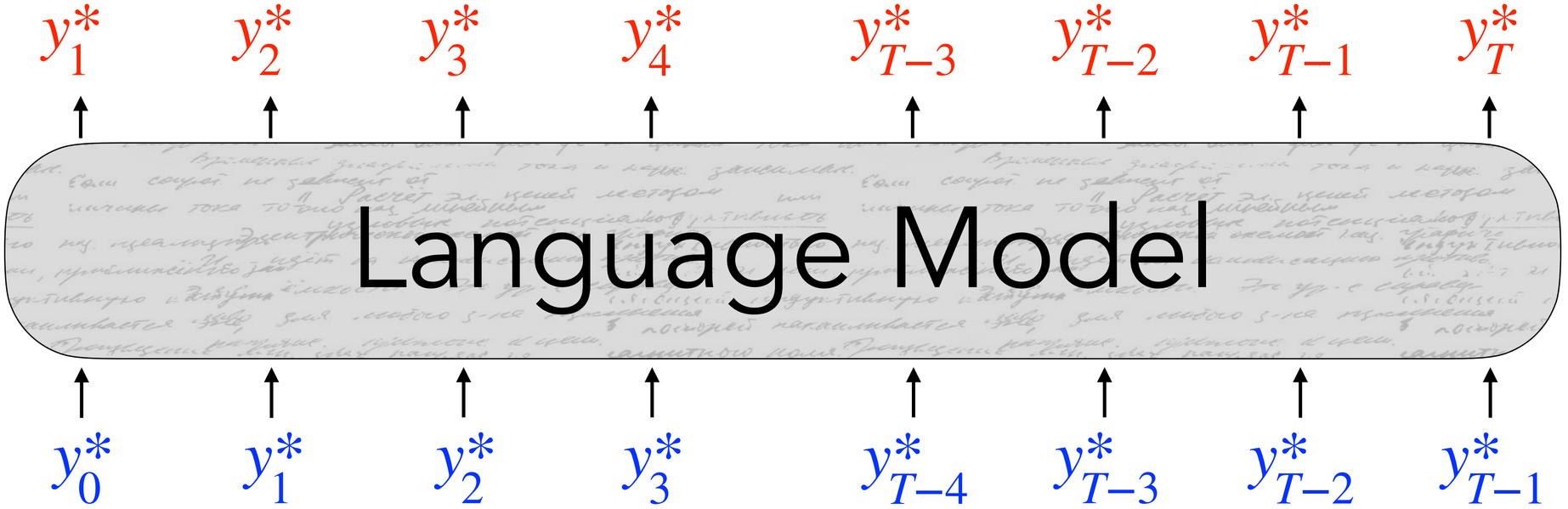
- Trained to generate the next word given a set of preceding words



Maximizing Likelihood

- Trained to generate the next word given a set of preceding words

$$\mathscr{L} = -\sum_{t=1}^{T} \log P(y_t^* \mid \{y^*\}_{<t})$$



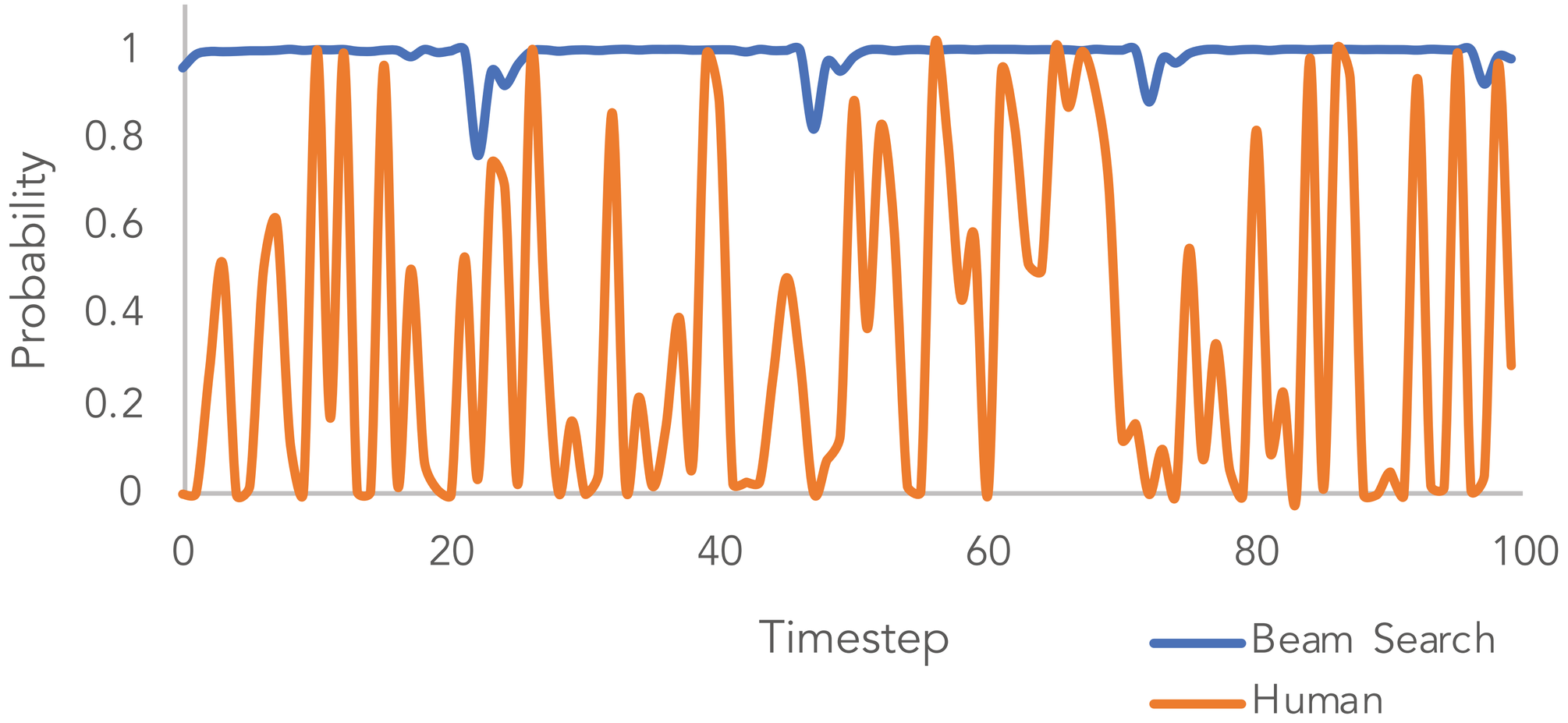
Issue #1: MLE *discourages* diversity

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Continuation: The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the **Universidad Nacional Autónoma de México (UNAM)** and **the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México...**

Holtzman et. al., ICLR 2020

Issue #1: MLE *discourages* diversity



Holtzman et. al., ICLR 2020

Alternatives: Unlikelihood Training

- Sequence-level Unlikelihood Training

- Given a set of undesired tokens $\mathscr{C}$, lower their likelihood in context

$$\mathscr{L}^t_{MLE} = -\log P(y^*_t \mid \{y^*\}_{<t}) \qquad \mathscr{L}^t_{UL} = -\sum_{y_{neg} \in \mathscr{C}} \log(1 - P(y_{neg} \mid \{y^*\}_{<t}))$$

Welleck et. al., ICLR 2020

Alternatives: Unlikelihood Training

- Sequence-level Unlikelihood Training

- Given a set of undesired tokens $\mathscr{C}$, lower their likelihood in context

$$\mathscr{L}^t_{MLE} = -\log P(y^*_t \mid \{y^*\}_{<t})$$

$$\mathscr{L}^t_{UL} = -\sum_{y_{neg} \in \mathscr{C}} \log(1 - P(y_{neg} \mid \{y^*\}_{<t}))$$

Typical Negative
Loglikehood objective

Welleck et. al., ICLR 2020

Alternatives: Unlikelihood Training

- Sequence-level Unlikelihood Training

- Given a set of undesired tokens $\mathscr{C}$, lower their likelihood in context

$$\mathscr{L}^t_{MLE} = -\log P(y^*_t \mid \{y^*\}_{<t})$$

Typical Negative Loglikehood objective

$$\mathscr{L}^t_{UL} = -\sum_{y_{neg} \in \mathscr{C}} \log(1 - P(y_{neg} \mid \{y^*\}_{<t}))$$

Unlikelihood objective lowers the probability of certain tokens

Welleck et. al., ICLR 2020

Alternatives: Unlikelihood Training

- Sequence-level Unlikelihood Training

- Given a set of undesired tokens $\mathscr{C}$, lower their likelihood in context

$$\mathscr{L}^t_{MLE} = -\log P(y^*_t \mid \{y^*\}_{<t}) \qquad \mathscr{L}^t_{UL} = -\sum_{y_{neg} \in \mathscr{C}} \log(1 - P(y_{neg} \mid \{y^*\}_{<t}))$$

$$\mathscr{L}^t_{ULE} = \mathscr{L}^t_{MLE} + \alpha \mathscr{L}^t_{UL}$$

Combine them for full unlikelihood training

Welleck et. al., ICLR 2020

Alternatives: Unlikelihood Training

- Sequence-level Unlikelihood Training

- Given a set of undesired tokens $\mathscr{C}$, lower their likelihood in context

$$\mathscr{L}^t_{MLE} = -\log P(y^*_t \mid \{y^*\}_{<t})$$

$$\mathscr{L}^t_{UL} = -\sum_{y_{neg} \in \mathscr{C}} \log(1 - P(y_{neg} \mid \{y^*\}_{<t}))$$

$$\mathscr{L}^t_{ULE} = \mathscr{L}^t_{MLE} + \alpha \mathscr{L}^t_{UL}$$

But wait, what's $\mathscr{C}$?

Welleck et. al., ICLR 2020

Alternatives: Unlikelihood Training

- Sequence-level Unlikelihood Training

- Given a set of undesired tokens $\mathscr{C}$, lower their likelihood in context

$$\mathscr{L}^t_{MLE} = -\log P(y^*_t \mid \{y^*\}_{<t})$$

$$\mathscr{L}^t_{UL} = -\sum_{y_{neg} \in \mathscr{C}} \log(1 - P(y_{neg} \mid \{y^*\}_{<t}))$$

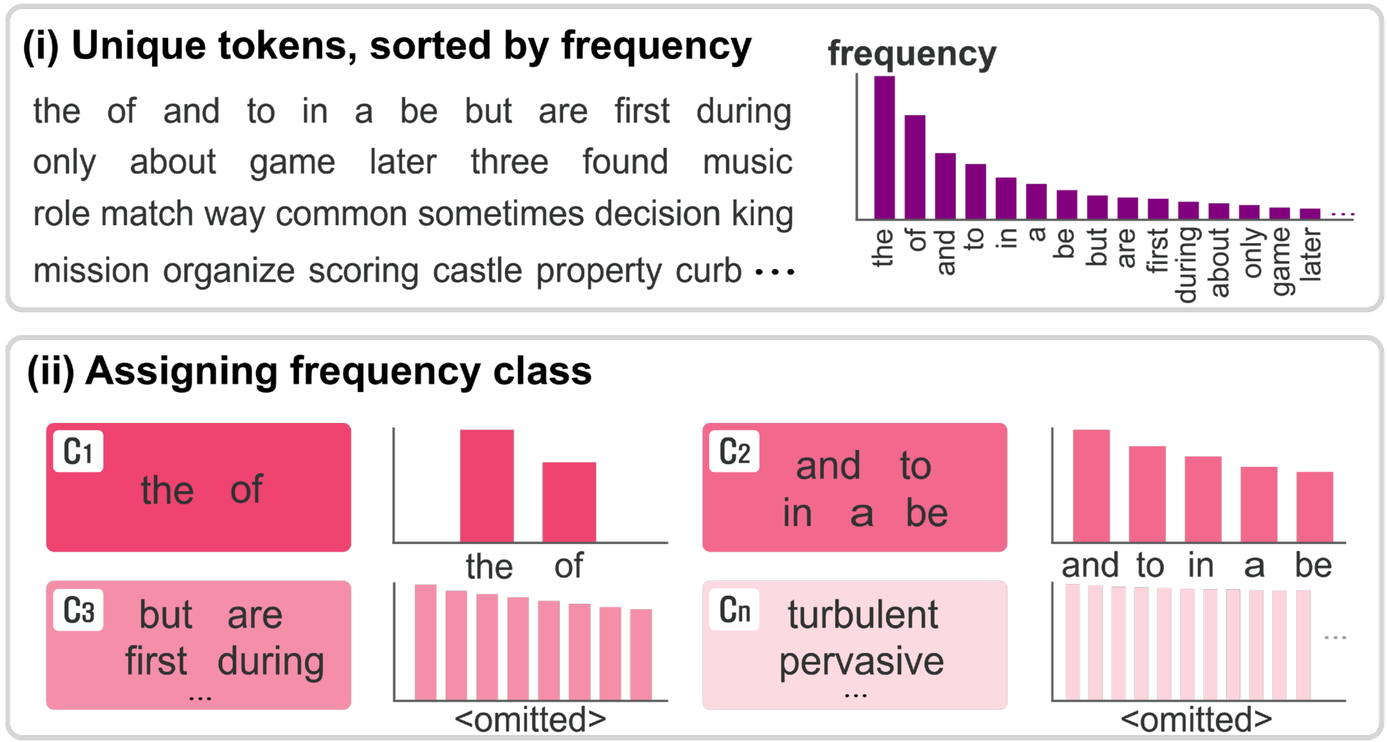
$$\mathscr{L}^t_{ULE} = \mathscr{L}^t_{MLE} + \alpha \mathscr{L}^t_{UL}$$

$$\mathscr{C} = \{y^*\}_{<t}$$

Welleck et. al., ICLR 2020

Alternatives: F^2 Softmax

- Avoid likelihood issues by factorizing the softmax
- Initialize *C* frequency classes
- Distribute vocabulary into classes so that token frequency uniformly distributed across **and** between classes



Choi et. al., EMNLP 2020

Alternatives: F^2 Softmax

- Learn to select both frequency class and vocabulary token during training

$$P(y_t = w_n \mid \{y\}_{<t}) = \frac{e^{U_n h}}{\sum_{m=1}^{M} e^{U_m h}}$$

$$P(y_t = w_n \mid \{y\}_{<t}) = \left(\frac{e^{V_f h}}{\sum_{c=1}^{C} e^{V_c h}} \right) \left(\frac{e^{U_n h}}{\sum_{m=1}^{M_f} e^{U_m h}} \right)$$

Issue #2: Exposure Bias

- During training, we condition on gold context tokens that are real human-generated text

$$\mathscr{L}_{MLE} = -\log P(y_t^* \mid \{y^*\}_{<t})$$

- During inference, we decode from distributions conditioned on previously generated tokens

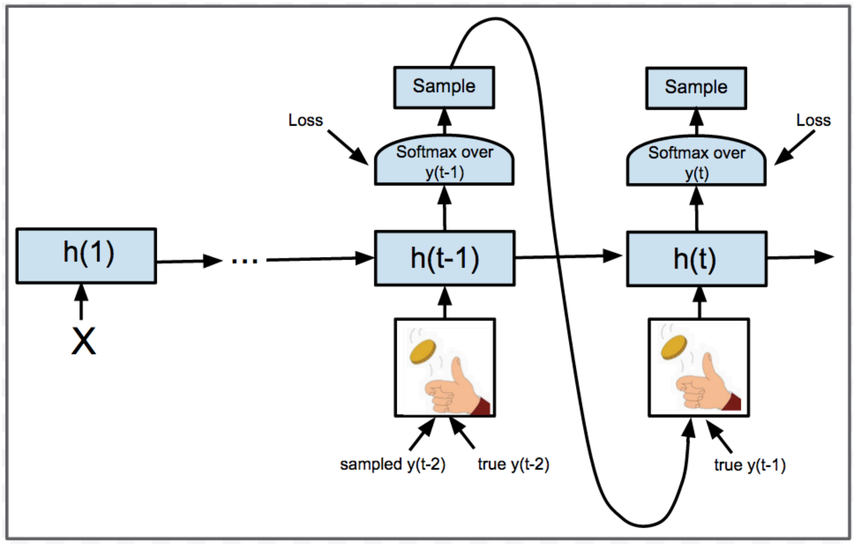
$$\mathscr{L}_{dec} = -\log P(\hat{y}_t \mid \{\hat{y}\}_{<t})$$

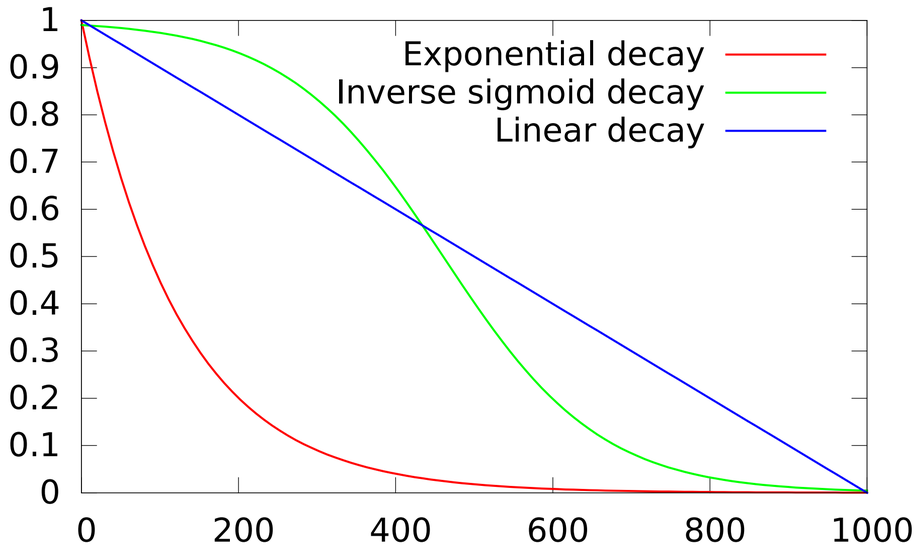
Alternative: Online Learning!

- Scheduled Sampling
- Dataset Aggregation (DAGGER)
- Reinforcement Learning
- Adversarial Learning

Alternative: Scheduled Sampling

- Barely used in newer transformer architectures, but relevant for RNNs
- After enough epochs of training with maximum likelihood estimation, interleave gold tokens and sampled tokens





Bengio et. al., NeurIPS 2015

Dataset Aggregation

1. Sample sequences according to your model
2. Add these sequences back to your training set
3. Re-train your model
4. Repeat

Ross et. al., AISTATS 2011

REINFORCE

- Sample sequences
- Use REINFORCE trick to scale losses by reward
- Define scoring metric as reward
- Set appropriate baseline
- Mix with

Adversarial Learning

- Similar to REINFORCE, but using discriminator as the reward function
-

Li et. al., EMNLP 2017

Issue #3: Can we learn other behaviors?

- Can we tie discriminators to desired behaviors? Yes! If we can formalize as reward functions
- Cross-modal consistency (Ren et al., CVPR 2017)
- Simplicity (Zhang and Lapata, EMNLP 2017)
- Temporal consistency (Bosselut et al., NAACL 2018)
- Politeness (Tan et al., TACL 2018)
- Paraphrasing (Li et al., EMNLP 2018)
- Sentiment (Gong et al., NAACL 2019)
- Formality (Gong et al., NAACL 2019)