Natural Language Generation:

Decoding & Training

Antoine Bosselut

EPFL

Today's Outline

- **Lecture:**
  - **Greedy Decoding**: Argmax, Beam Search
  - **Sampling Methods:** Top-k, Top-p
  - **Training Challenges:** Exposure bias, Reinforcement Learning
- **A2 Q&A Session**
- **Tomorrow:**
  - **Lecture:** Text generation evaluation
  - **A2 Q&A Session**
  - **Review of Week 5 Exercise Session:** Robustness & Prompting
  - **Week 6 Exercise Session:** Text Generation

Decoding: what is it all about?

- At each time step *t,* our model computes a vector of scores for each token in our vocabulary, $S \in \mathbb{R}^V$:

$$S = f\left(\{y_{<t}\}\right)$$

$f(.)$ is your model

- Then, we compute a probability distribution P over these scores (usually with a softmax function):

$$P\left(y_t = w \,\middle|\, \{y_{<t}\}\right) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- Our decoding algorithm defines a function to select a token from this distribution:

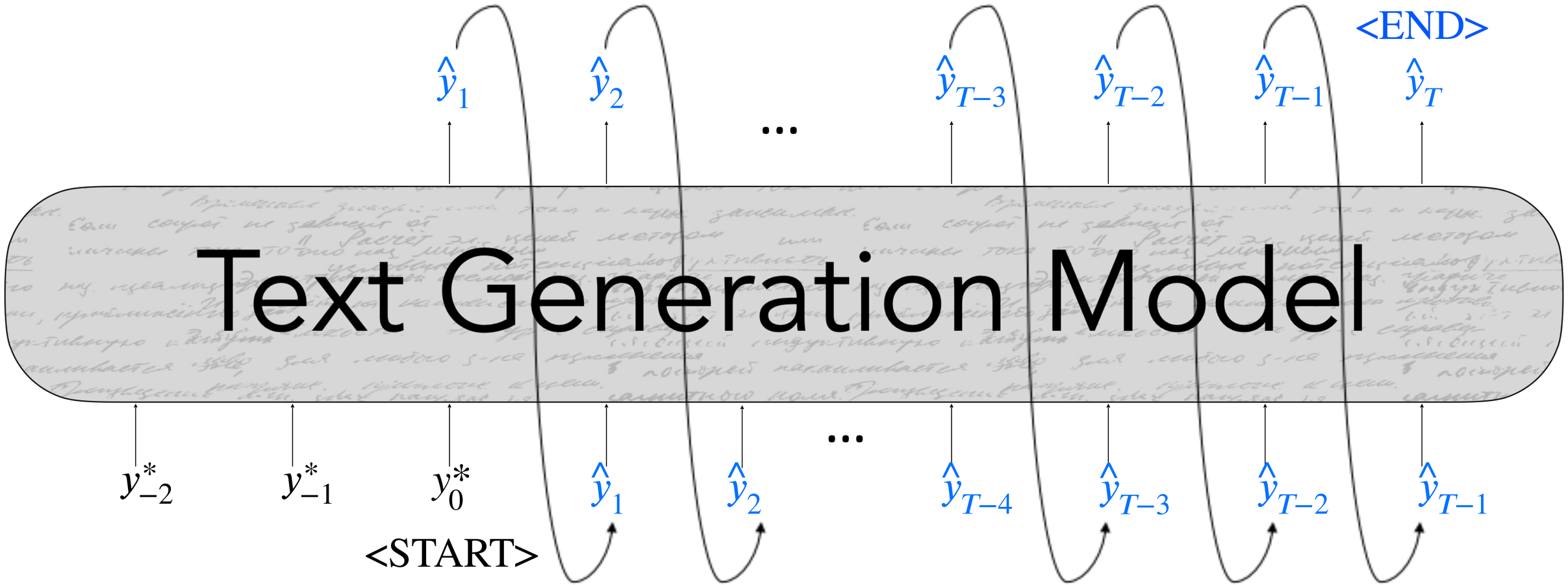
$$\hat{y}_t = g\left(P\left(y_t \,|\, \{y_{<t}\}\right)\right)$$

$g(.)$ is your decoding algorithm

Decoding: what is it all about?

- Our decoding algorithm defines a function to select a token from this distribution

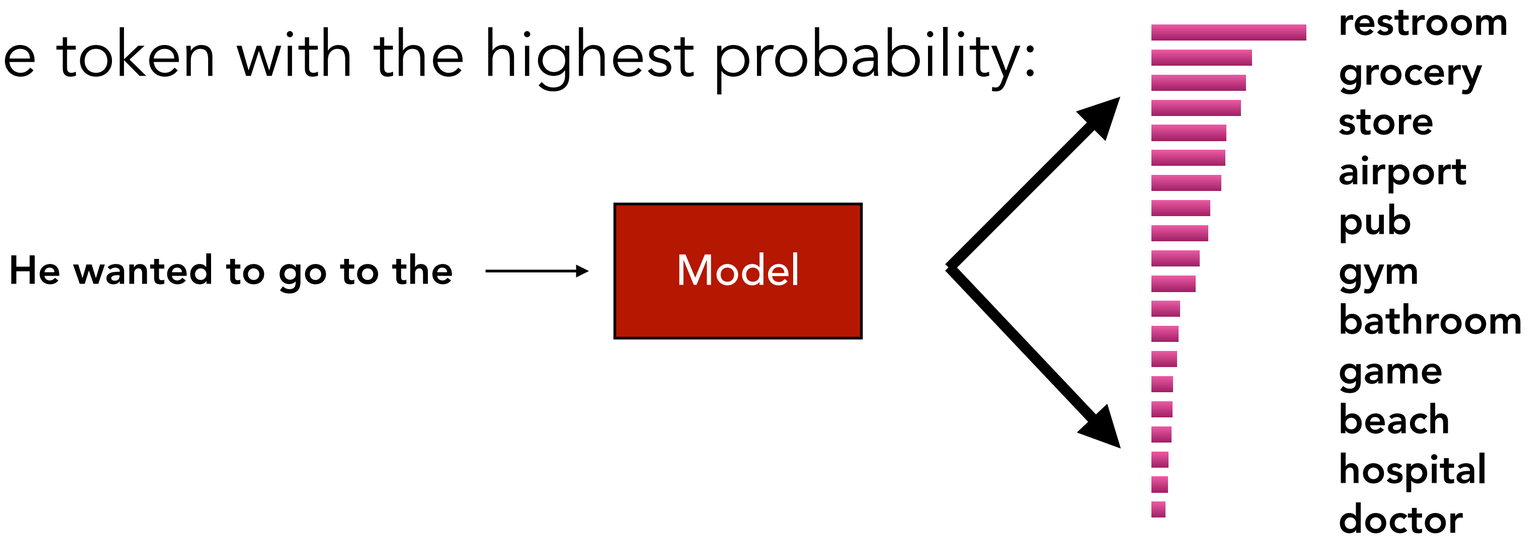
$$\hat{y}_t = g\left(P\left(y_t \mid \{y^*\}, \hat{y}_{<t}\right)\right)$$



Greedy methods: Argmax Decoding

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

- g = select the token with the highest probability:

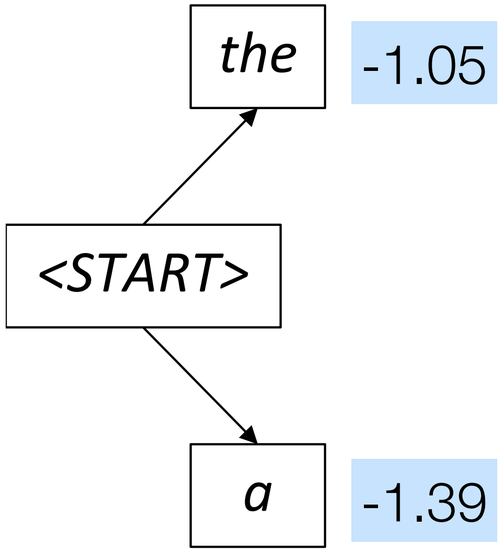


Greedy methods: Beam Search

- In greedy decoding, we cannot revise prior decisions
  - *les pauvres sont démunis (the poor don't have any money)*
  - → *the* ____
  - → *the poor* ____
  - → *the poor are* ____
- Beam Search: Explore several different hypotheses instead of just one
  - Keep track of the the *b* most probable sequences at each decoder step instead of just one
  - *b* is called the **beam size**

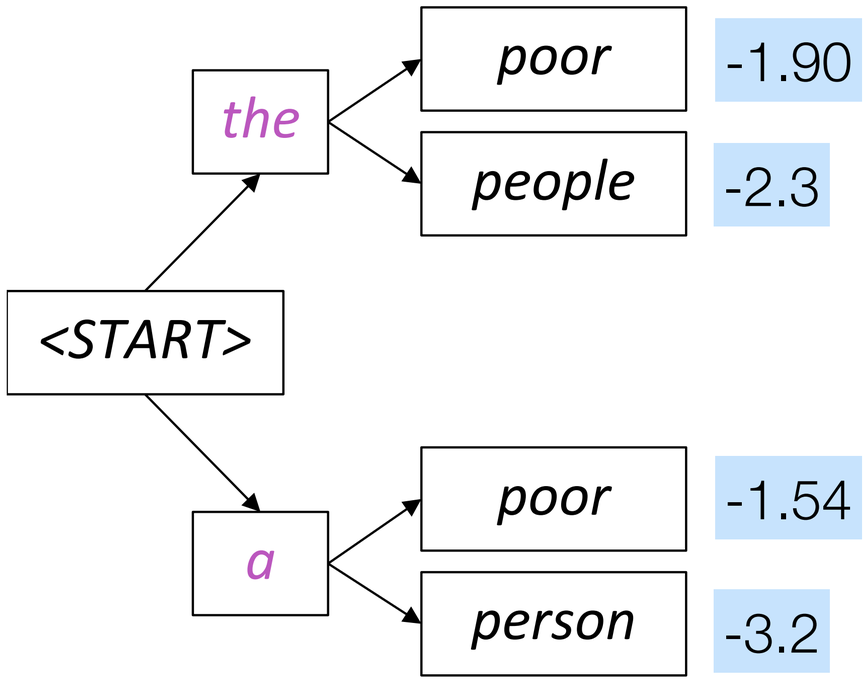
Greedy methods: Beam Search

Beam size = 2



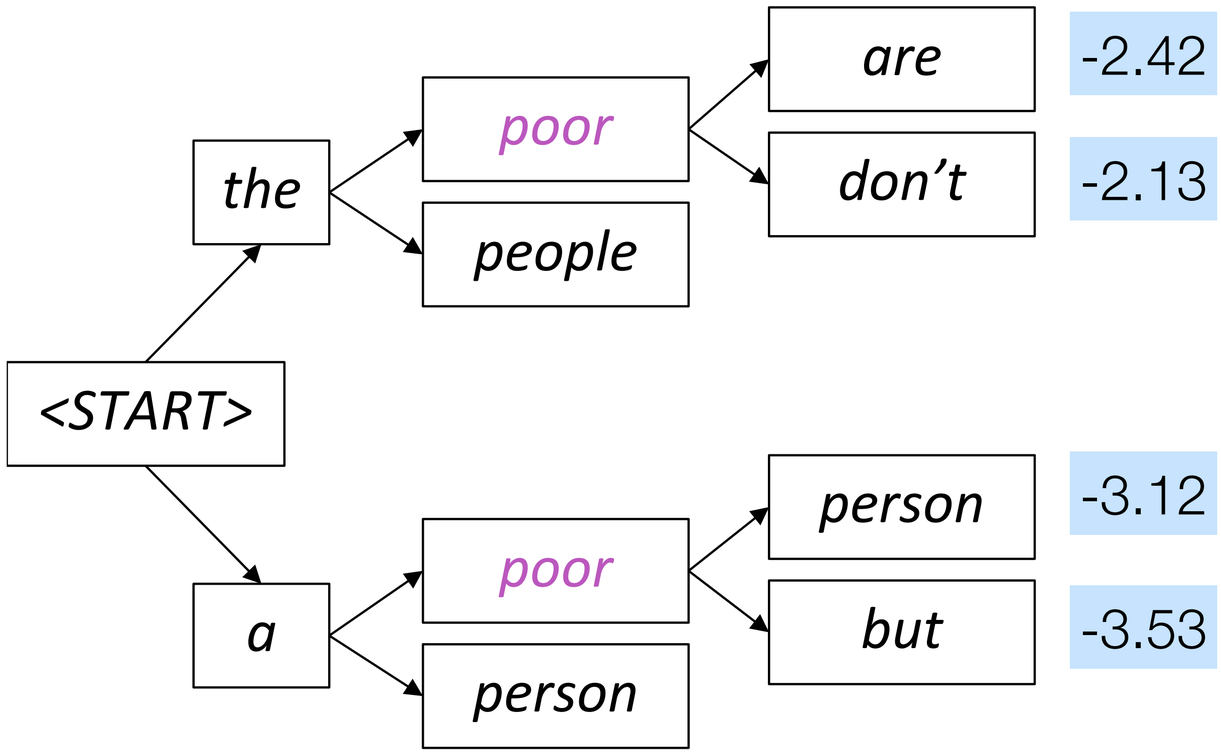
Greedy methods: Beam Search

Beam size = 2



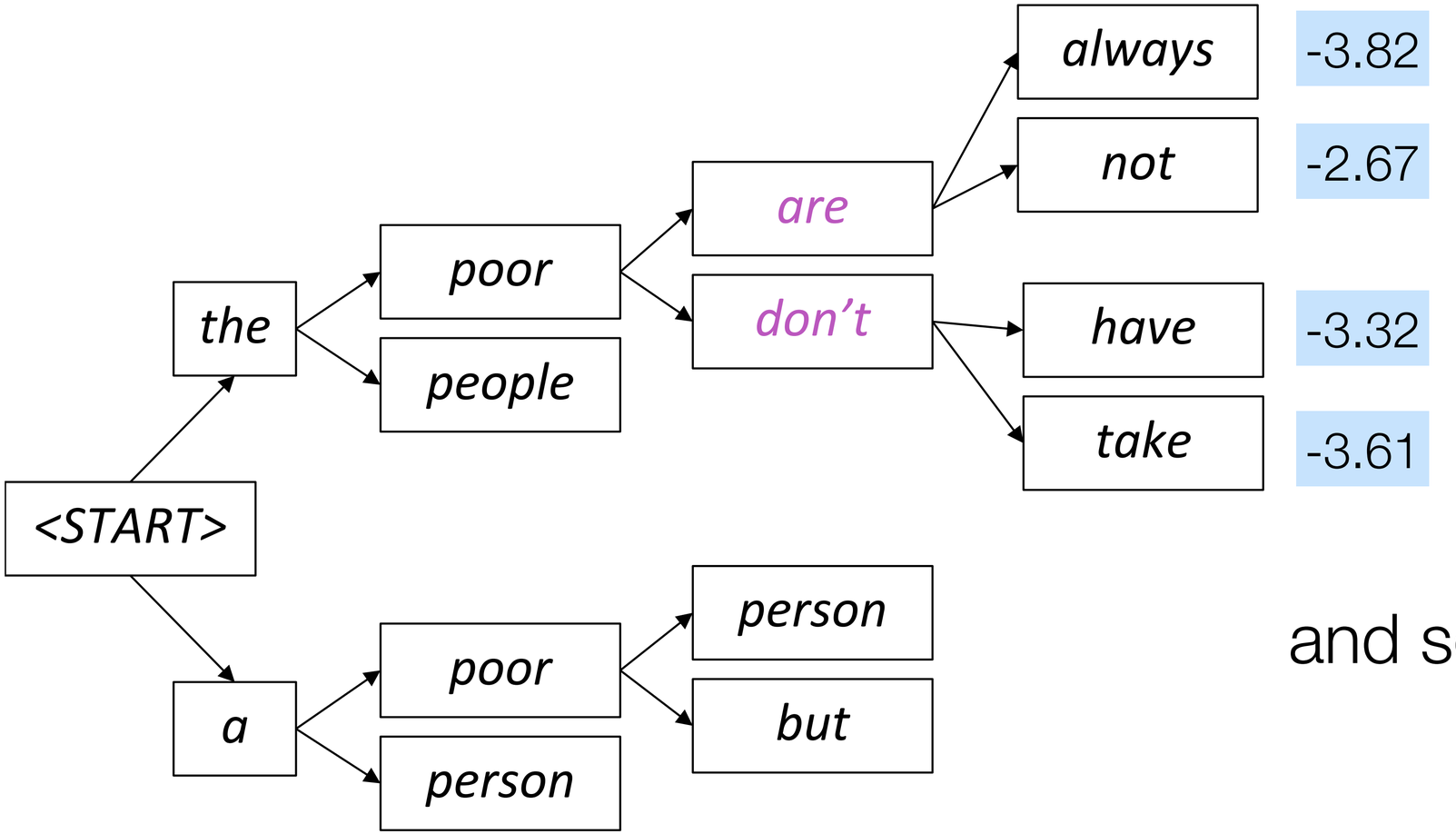
Greedy methods: Beam Search

Beam size = 2



Greedy methods: Beam Search

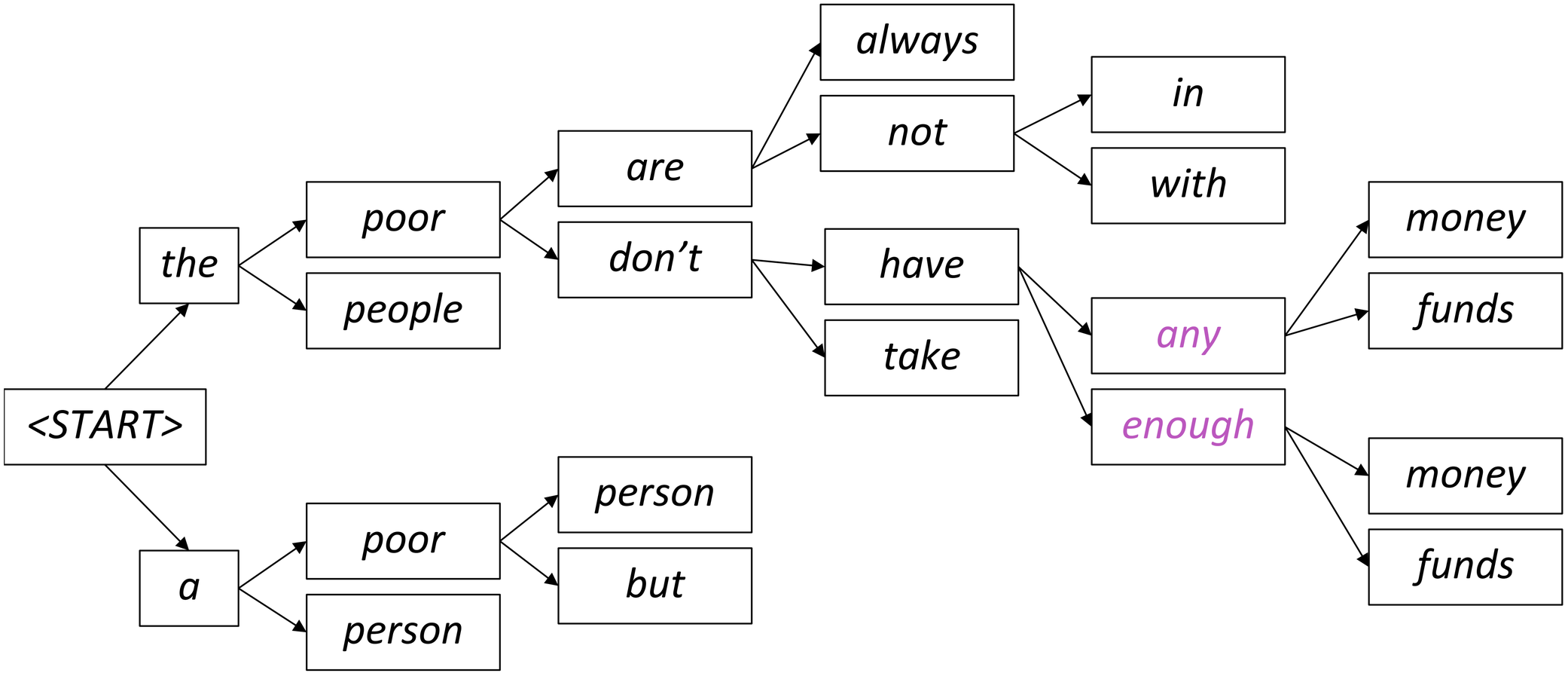
Beam size = 2



and so on…

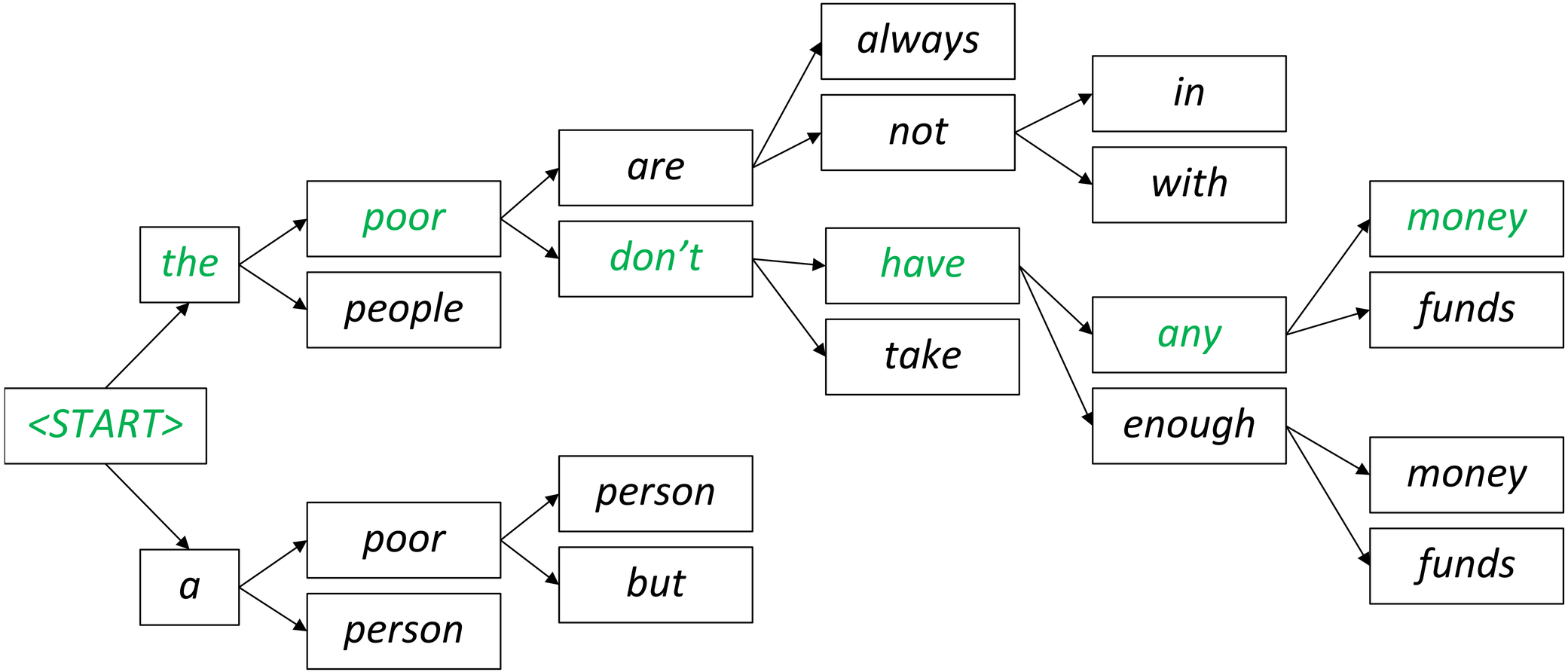
Greedy methods: Beam Search

Beam size = 2



Greedy methods: Beam Search

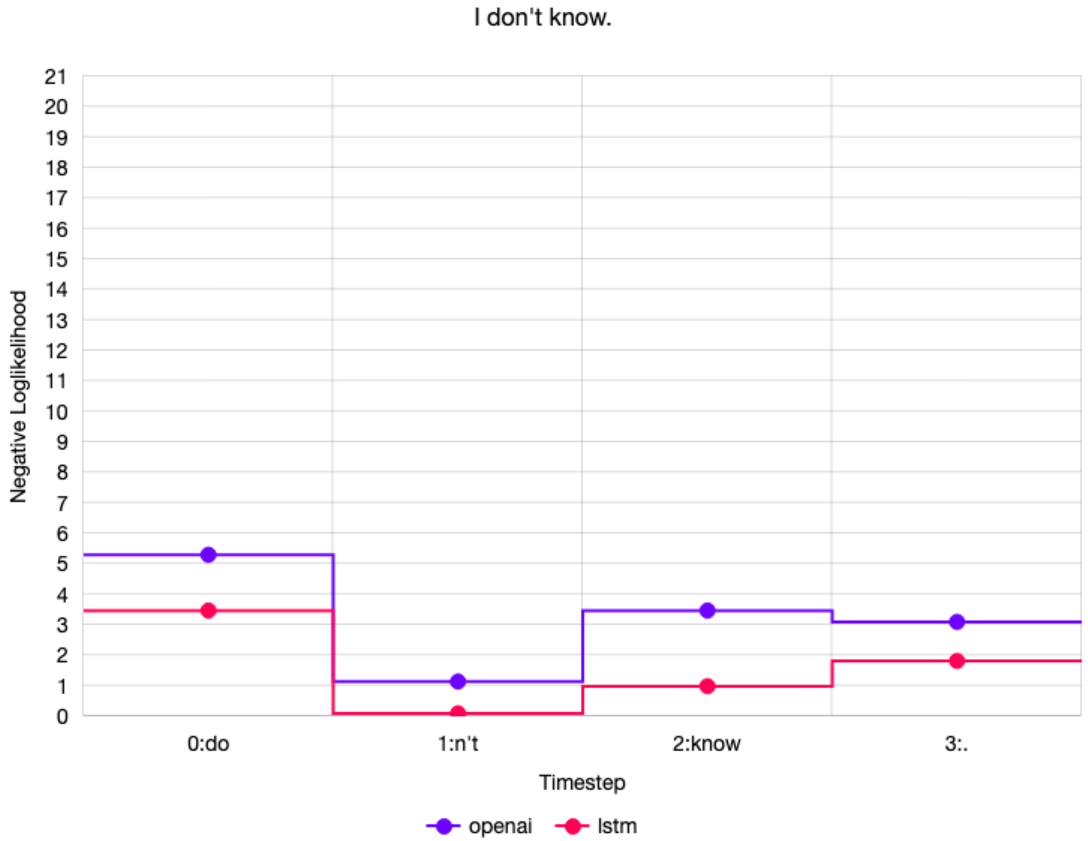
Beam size = 2



What do you think might be an inherent problem with greedy decoding?

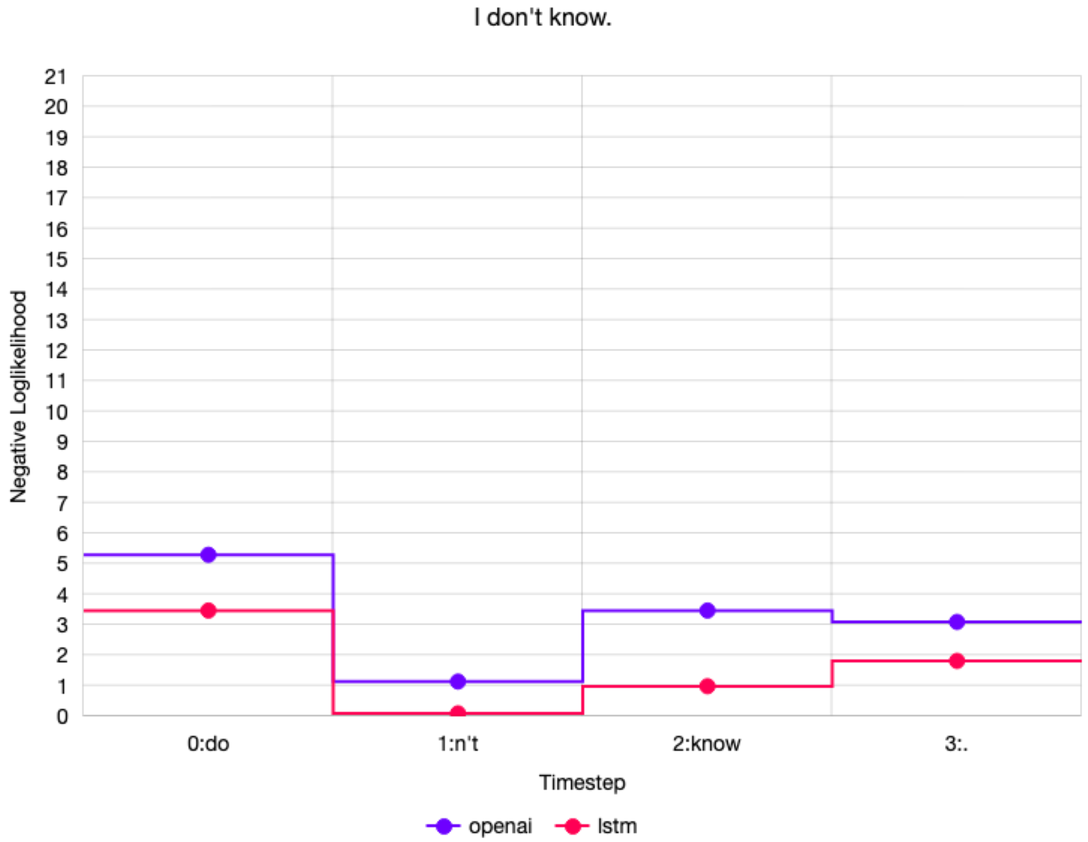
They maximise the likelihood of the sequence.
What do maximum likelihood sequences look like?

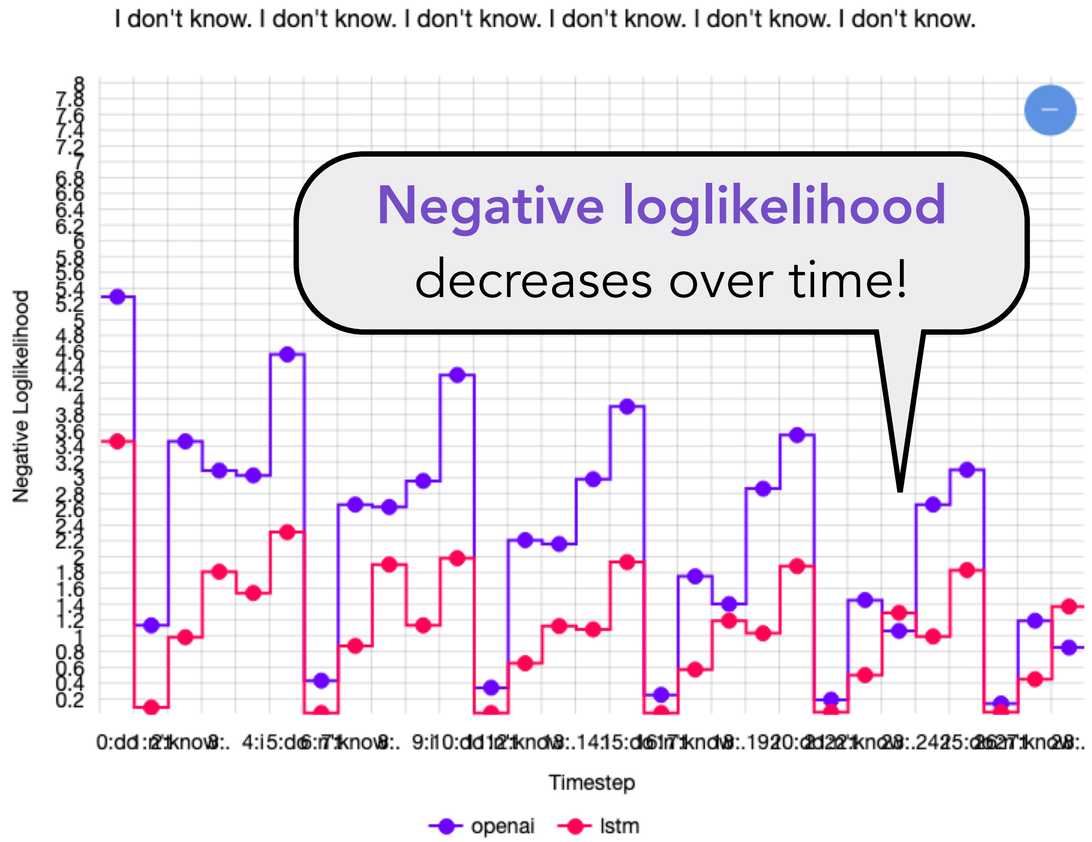
Step-by-step NLLs



(Holtzman et. al., ICLR 2020)

Step-by-step NLLs

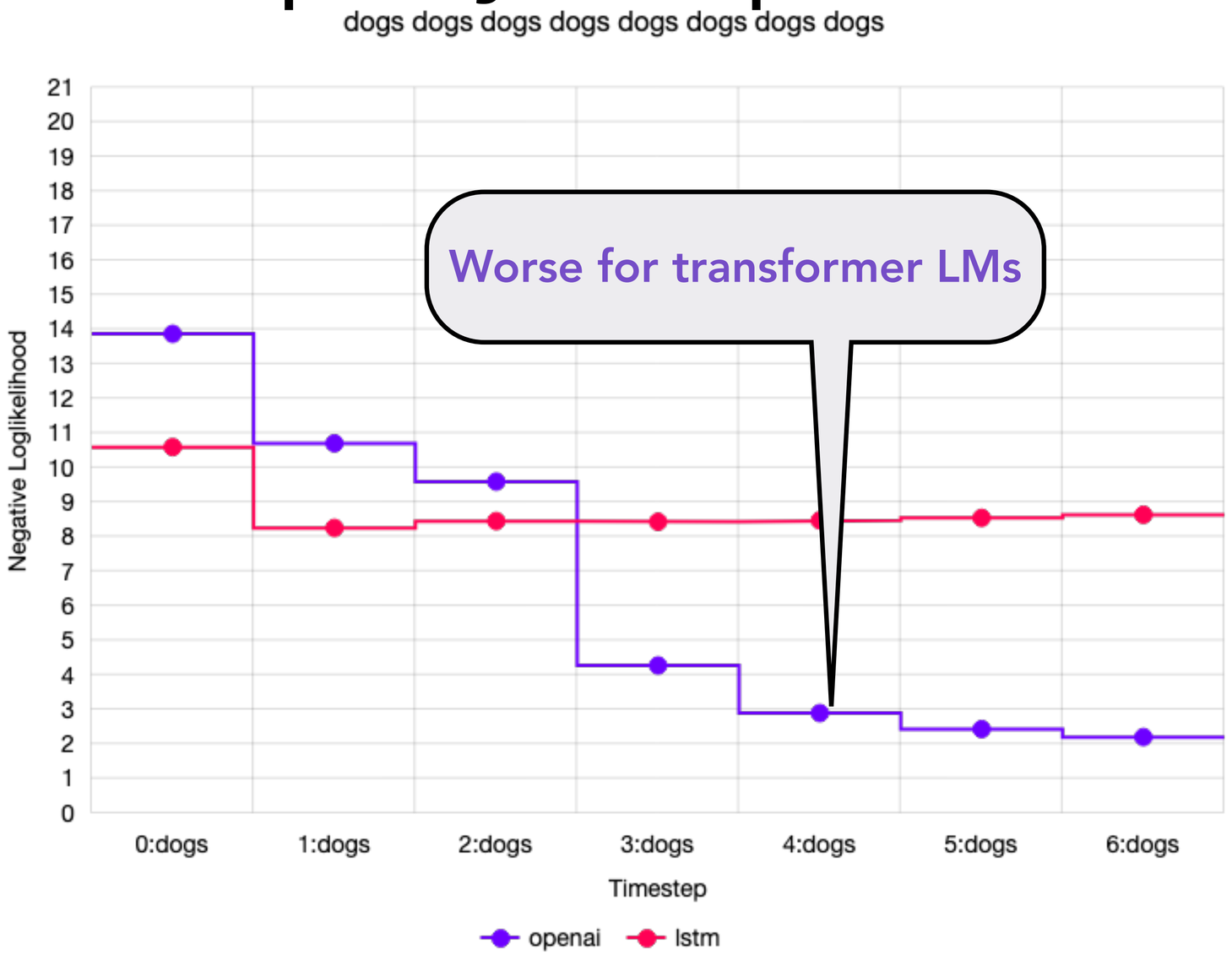




What do you notice as the number of time steps increases?

(Holtzman et. al., ICLR 2020)

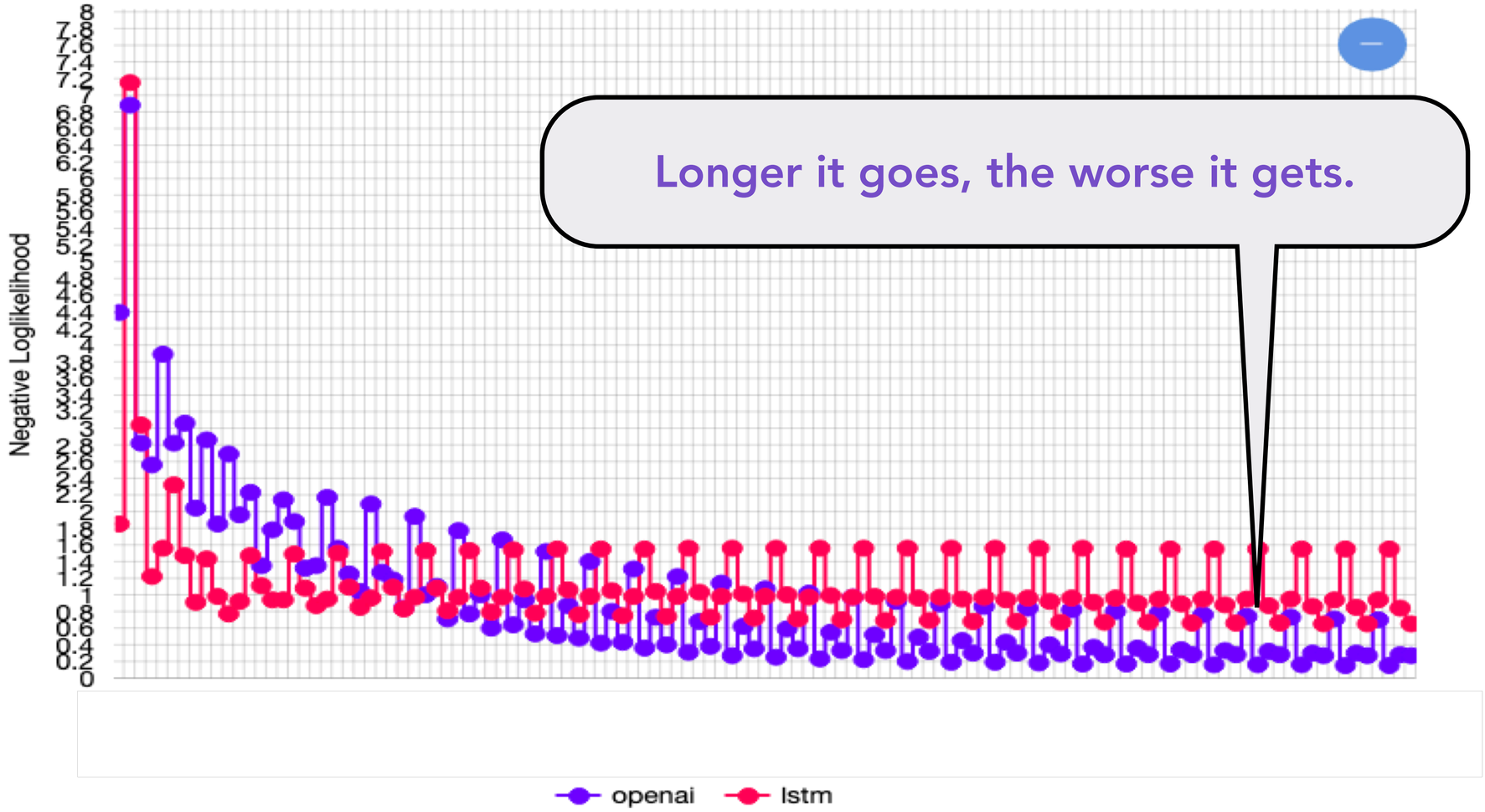
Step-by-step NLLs



(Holtzman et. al., ICLR 2020)

And it keeps going…

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)

Greedy methods get 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)

Greedy methods get 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.

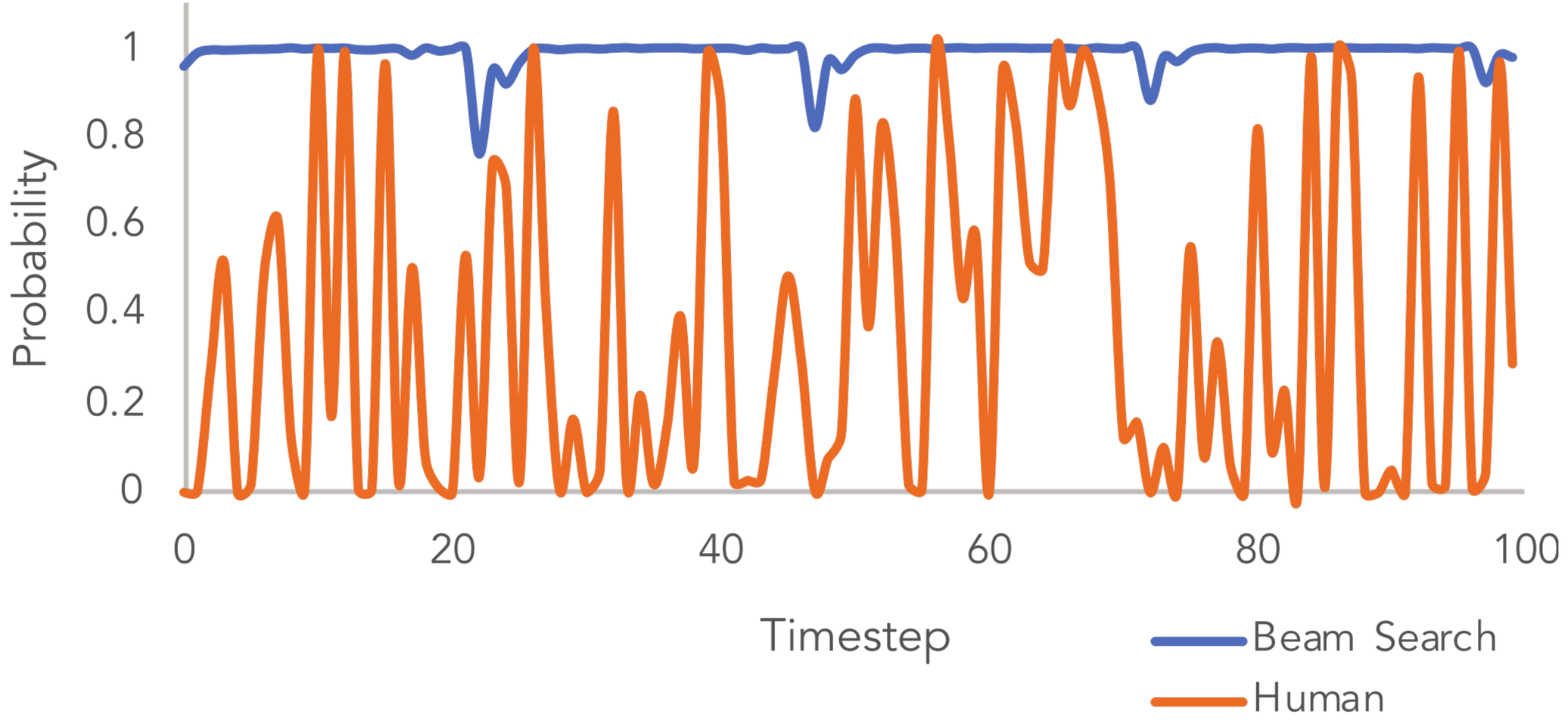
Continua

Repetition is a big problem in text generation!

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)

Are greedy methods reasonable?



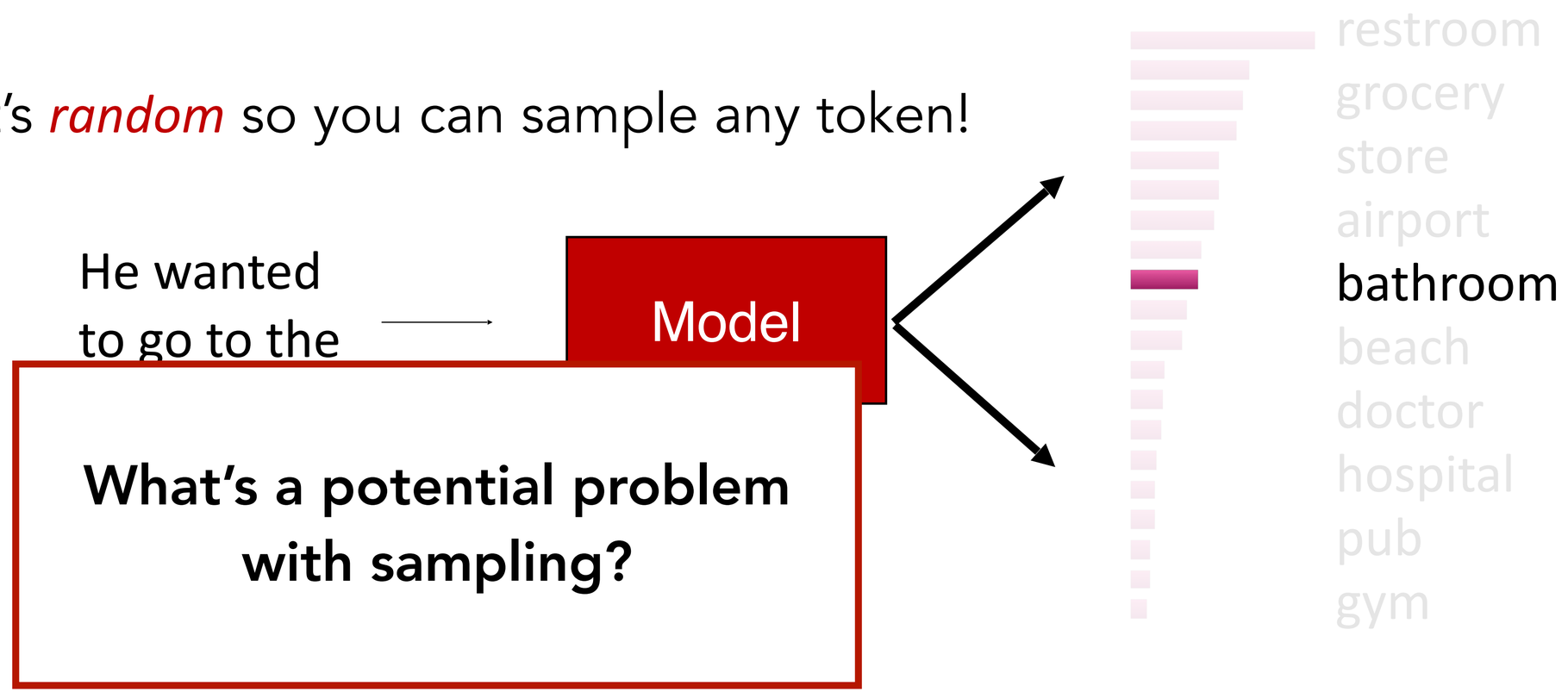
(Holtzman et. al., ICLR 2020)

Time to get *random* : Sampling!

- Sample a token from the distribution of tokens

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

- It's *random* so you can sample any token!



Decoding: Top-*k* sampling

- Problem: Vanilla sampling makes every token in the vocabulary an option
  - Even if most of the probability mass in the distribution is over a limited set of options, the **tail of the distribution could be very long**
  - Many tokens are probably irrelevant in the current context
  - Why are we giving them *individually* a tiny chance to be selected?
  - Why are we giving them *as a group* a high chance to be selected?

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

Decoding: Top-*k* sampling

- Problem: Vanilla sampling makes every token in the vocabulary an option
  - Even if most of the probability mass in the distribution is over a limited set of options, the **tail of the distribution could be very long**
  - Many tokens are probably irrelevant in the current context
  - Why are we giving them *individually* a tiny chance to be selected?
  - Why are we giving them *as a group* a high chance to be selected?

- Solution: Top-*k* sampling
  - Only sample from the top *k* tokens in the probability distribution

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

Decoding: Top-*k* sampling

- <u>Solution:</u> Top-*k* sampling
  - Only sample from the top *k* tokens in the probability distribution
  - Common values are *k* = 5, 10, 20 (*but it's up to you!*)

He wanted to go to the → Model

  - Increase *k* for more **diverse**/**risky** outputs
  - Decrease *k* for more **generic**/**safe** outputs

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

Decoding: Top-*k* sampling

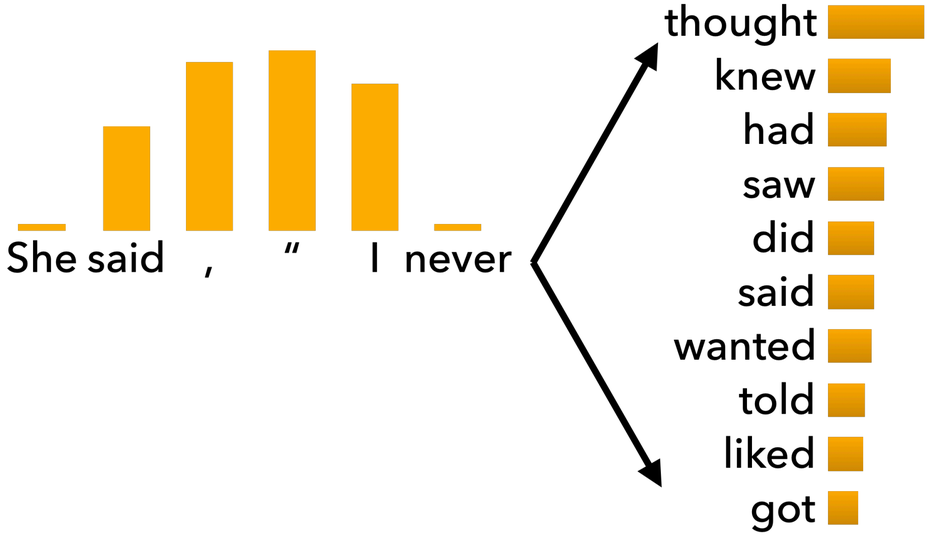
- <u>Solution:</u> Top-*k* sampling
  - Only sample from the top *k* tokens in the probability distribution
  - Common values are *k* = 5, 10, 20 (*but it's up to you!*)

What's a potential problem with top-*k* sampling?

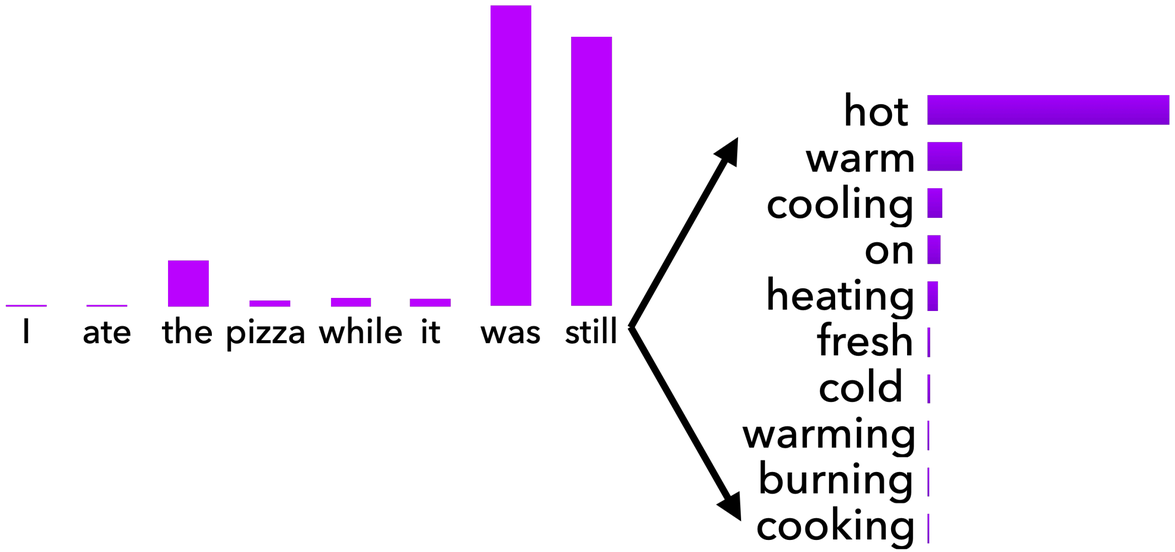
  - Increase *k* for more **diverse**/**risky** outputs
  - Decrease *k* for more **generic**/**safe** outputs

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

Issues with Top-*k* sampling



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



Top-*k* sampling can also cut off too ***slowly***!
(same issue as vanilla sampling!)

(Holtzman et. al., ICLR 2020)

Decoding: Top-*p* (nucleus) sampling

- Problem: The probability distributions we sample from are dynamic
  - When the distribution P_t is flatter, a limited *k* removes many viable options
  - When the distribution P_t is peakier, a high *k* allows for too many options to have a chance of being selected

(Holtzman et. al., ICLR 2020)

Decoding: Top-*p* (nucleus) sampling

- Problem: The probability distributions we sample from are dynamic
  - When the distribution P_t is flatter, a limited *k* removes many viable options
  - When the distribution P_t is peakier, a high *k* allows for too many options to have a chance of being selected

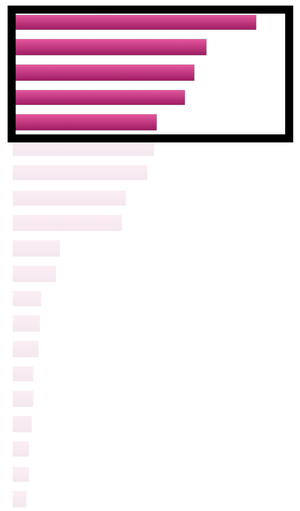
- Solution: Top-*p* sampling
  - Sample from all tokens in the top *p* cumulative probability mass (i.e., where mass is concentrated)
  - Varies *k* depending on the uniformity of P_t

(Holtzman et. al., ICLR 2020)

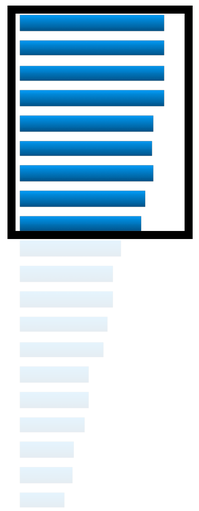
Decoding: Top-*p* (nucleus) sampling

- <u>Solution:</u> Top-*p* sampling
  - Sample from all tokens in the top *p* cumulative probability mass (i.e., where mass is concentrated)
  - Varies *k* depending on the uniformity of P_t

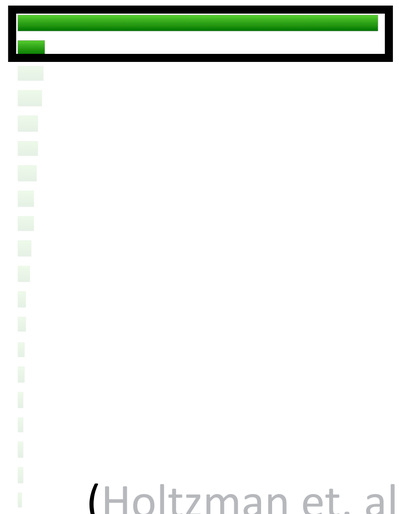
$$P_t^1\left(y_t = w \mid \{y\}_{<t}\right)$$



$$P_t^2\left(y_t = w \mid \{y\}_{<t}\right)$$



$$P_t^3\left(y_t = w \mid \{y\}_{<t}\right)$$



(Holtzman et. al., ICLR 2020)

Scaling randomness: Softmax temperature

- **Recall**: At every time step, the model computes a probability distribution by applying the softmax function to a vector of scores

$$P_t(y_t = w) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- Apply a temperature hyperparameter to the softmax to rebalance :

$$P_t\left(y_t = w\right) = \frac{\exp\left(S_w/\tau\right)}{\sum_{w' \in V} \exp\left(S_{w'}/\tau\right)}$$

What happens if we increase the temperature?

Scaling randomness: Softmax temperature

- **Recall**: At every time step, the model computes a probability distribution by applying the softmax function to a vector of scores

$$P_t(y_t = w) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- Apply a temperature hyperparameter to the softmax to rebalance :

$$P_t\left(y_t = w\right) = \frac{\exp\left(S_w/\tau\right)}{\sum_{w' \in V} \exp\left(S_{w'}/\tau\right)}$$

- Raise the temperature $\tau > 1$:
  - P_t becomes more uniform
  - More diverse output (probability is spread around vocabulary)

What happens if we decrease the temperature?

Scaling randomness: Softmax temperature

- **Recall**: At every time step, the model computes a probability distribution by applying the softmax function to a vector of scores

$$P_t(y_t = w) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- Apply a temperature hyperparameter to the softmax to rebalance :

$$P_t\left(y_t = w\right) = \frac{\exp\left(S_w/\tau\right)}{\sum_{w' \in V} \exp\left(S_{w'}/\tau\right)}$$

- Raise the temperature $\tau > 1$:
  - P_t becomes more uniform
  - More diverse output (probability is spread around vocabulary)
- Lower the temperature $\tau < 1$:
  - P_t becomes more spiky
  - Less diverse output (probability is concentrated on top words)

What happens if temperature goes to 0?

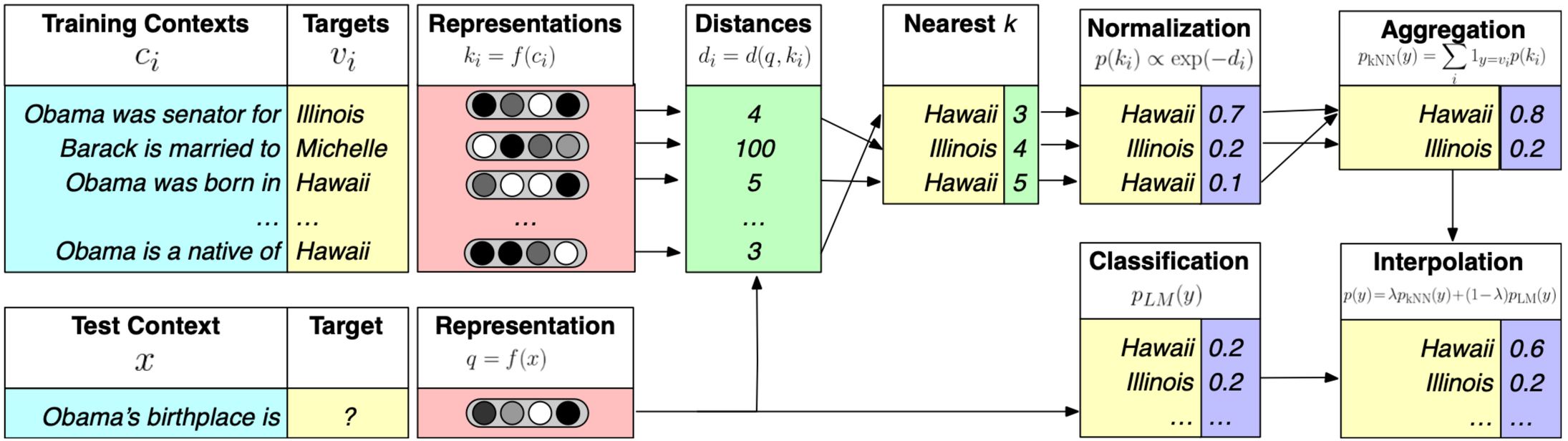
$$P_t(y_t = w) = \frac{\exp(S_w/\tau)}{\sum_{w' \in V} \exp(S_{w'}/\tau)}$$

All mass moves to highest probability token —> Greedy Decoding!

Improving decoding: re-balancing distributions

- Problem: What if I don't trust how well my model's distributions are calibrated?
  - Don't rely on **ONLY** your model's distribution over tokens

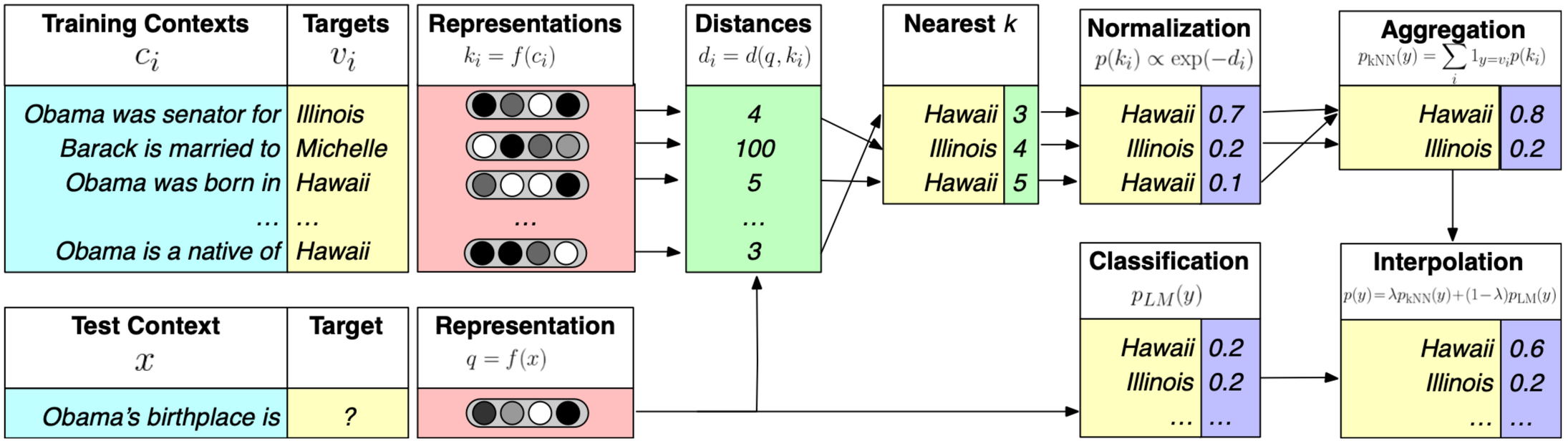
- Solution #1: Re-balance P_t using retrieval from n-gram phrase statistics!



(Khandelwal et. al., ICLR 2020)

Improving decoding: re-balancing distributions

- Solution #1: Re-balance P_t using retrieval from n-gram phrase statistics!
  - Cache a database of phrases from your training corpus (or some other corpus)
  - At decoding time, search for most similar phrases in the database
  - Re-balance P_t using induced distribution P_{phrase} over words that follow these phrases



(Khandelwal et. al., ICLR 2020)

Improving Decoding: Re-ranking

- **Problem**: What if I decode a bad sequence from my model?
- **Solution**: Decode a bunch of sequences
  - 10 candidates is a common number, but it's up to you
- Define a score to approximate quality of sequences and re-rank by this score
  - Simplest is to use perplexity!
    - Careful! Remember that **repetitive sequences** can get low perplexity.
  - Re-rankers can score a variety of properties:
    - style (Holtzman et al., 2018), discourse (Gabriel et al., 2021), entailment/factuality (Goyal et al., 2020), logical consistency (Lu et al., 2020), and many more…
    - **Beware of poorly-calibrated re-rankers**
  - Can use multiple re-rankers in parallel

Decoding: Takeaways

- Decoding is still a challenging problem in natural language generation
- Human language is not well-approximated by **probability maximization**
- Different decoding algorithms can inject biases that encourage different properties of coherent natural language generation
- Some of the most **impactful advances** in NLG of the last few years have come from **simple**, but **effective**, modifications to decoding algorithms
- **A lot more work to be done!**

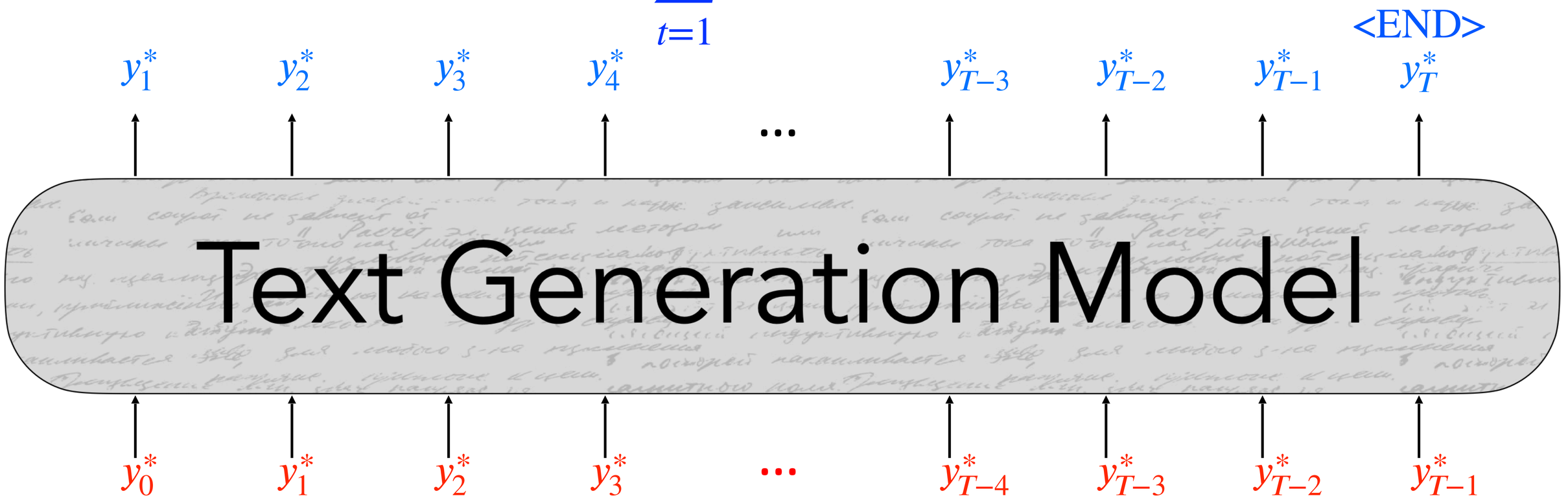
Why do we need these fancy decoders?

Shouldn’t our models just be better calibrated?

Maximum Likelihood Training (i.e., *teacher forcing*)

- Trained to generate the next word y_t^* given a set of preceding words $\{y^*\}_{<t}$

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



Issue #1: MLE *discourages* diversity

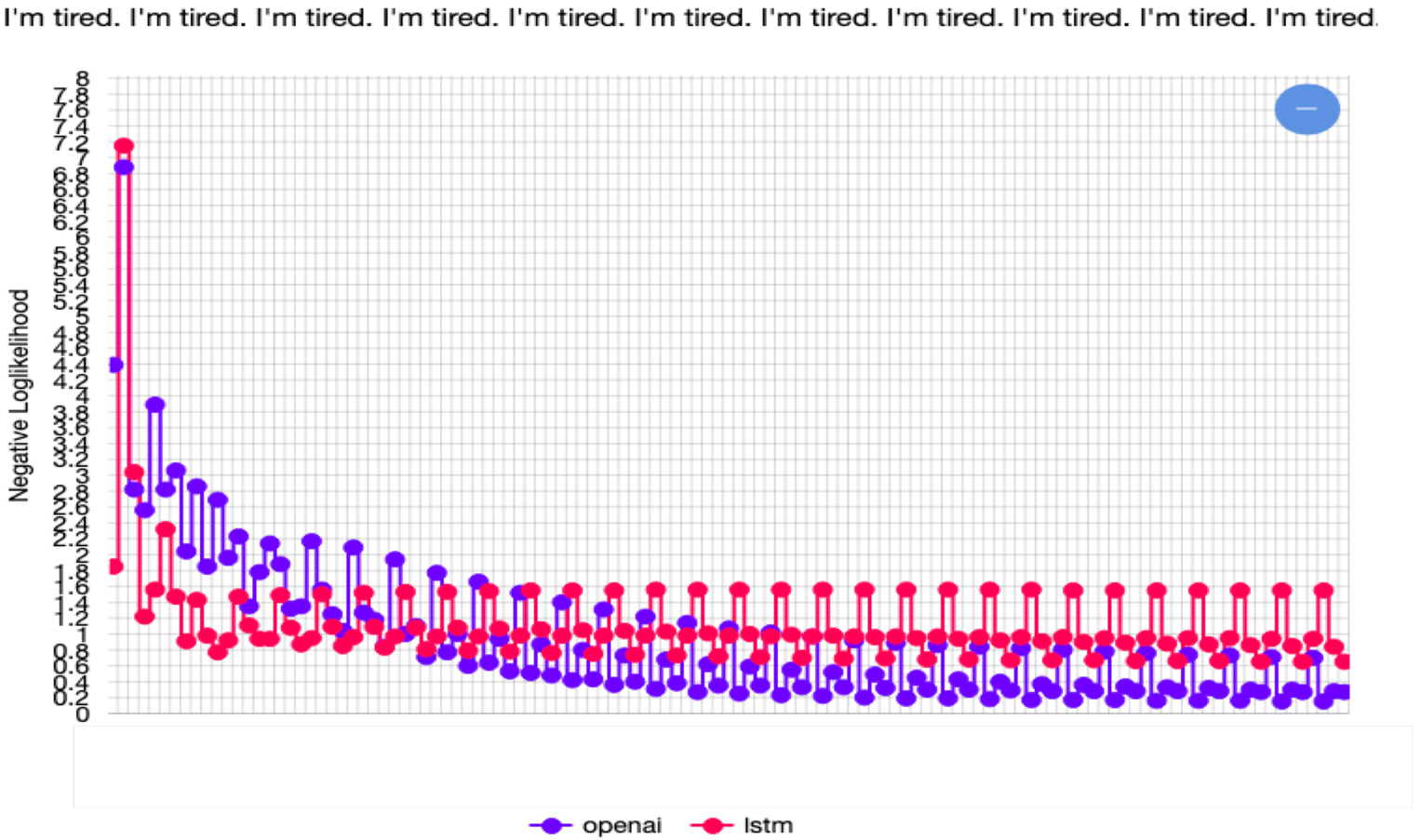
Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

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

- Maximum Likelihood Estimation *discourages* diverse text generation

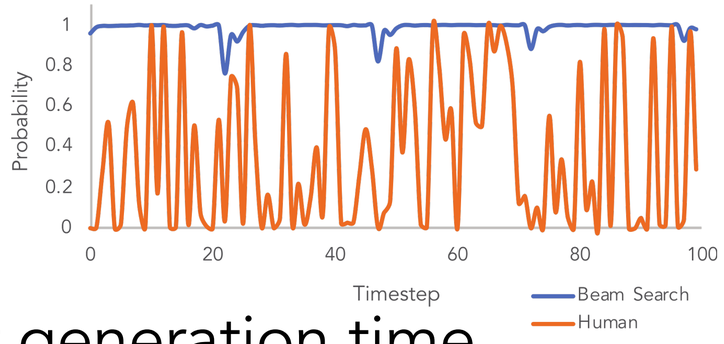


How can we increase the diversity of the sequences that are seen during training?

More data! Pretraining!

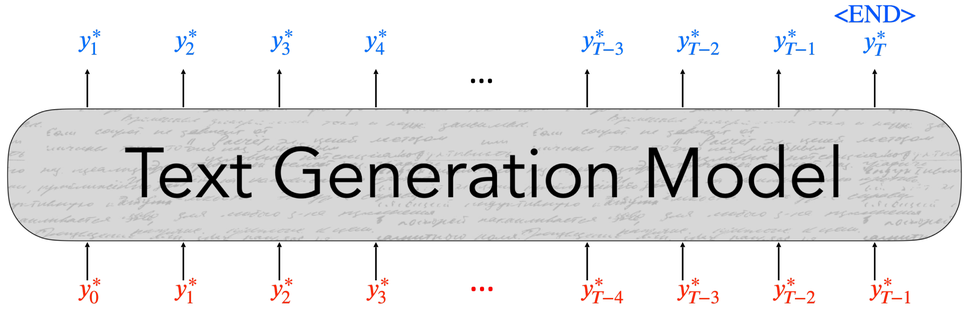
Learn from the sequences the model generates itself!

Issue #2: Exposure Bias



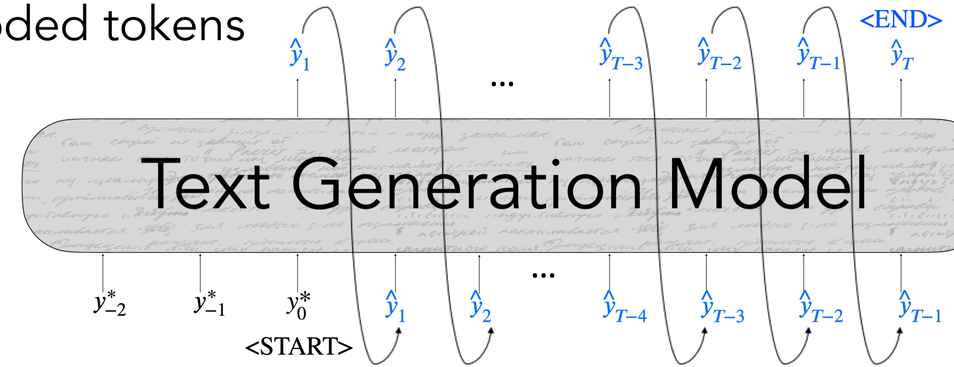
- Training with teacher forcing leads to *exposure bias* at generation time
  - During training, our model's inputs are gold context tokens from real, human-generated texts

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



  - At generation time, our model's inputs are previously–decoded tokens

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



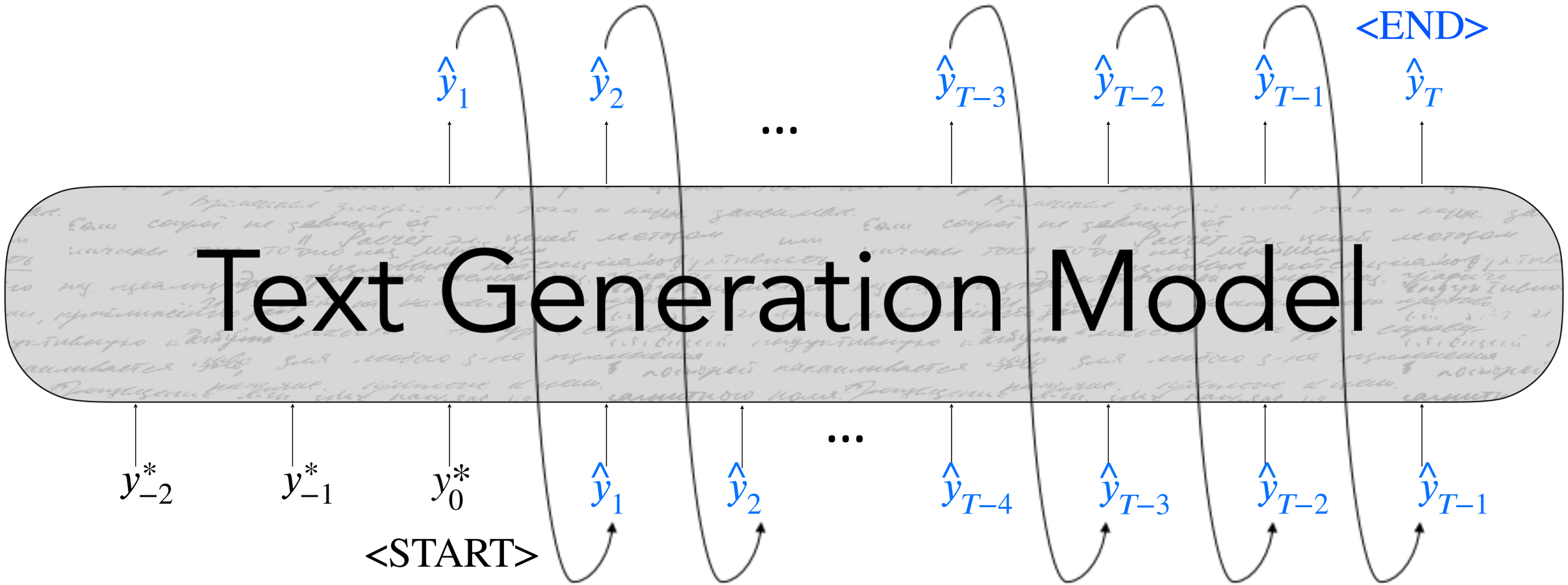
Solutions

- Reinforcement Learning: cast your text generation model as a Markov decision process
  - **State** *s* is the model's representation of the preceding context
  - **Actions** *a* are the words that can be generated
  - **Policy** π is the decoder
  - **Rewards** *r* are provided by an external score
  - Learn behaviors by rewarding the model when it exhibits them

REINFORCE: Basics

- Sample a sequence from your model

$$\mathscr{L}_{RL} = -\sum_{t=1}^{T} r(\hat{y}) \log P\left(\hat{y}_t \mid \{y^*\}; \{\hat{y}\}_{<t}\right)$$



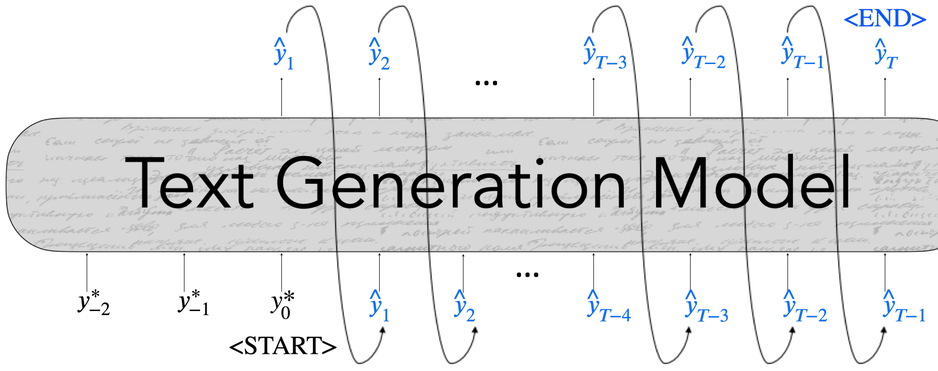
REINFORCE: Basics

- Sample a sequence from your model

Next time, increase the probability of this sampled token in the same context.

$$\mathscr{L}_{RL} = -\sum_{t=1}^{T} \log P\left(\hat{y}_t \mid \{y^*\}; \{\hat{y}\}_{<t}\right)$$

…but do it more if I get a high reward from the reward function.

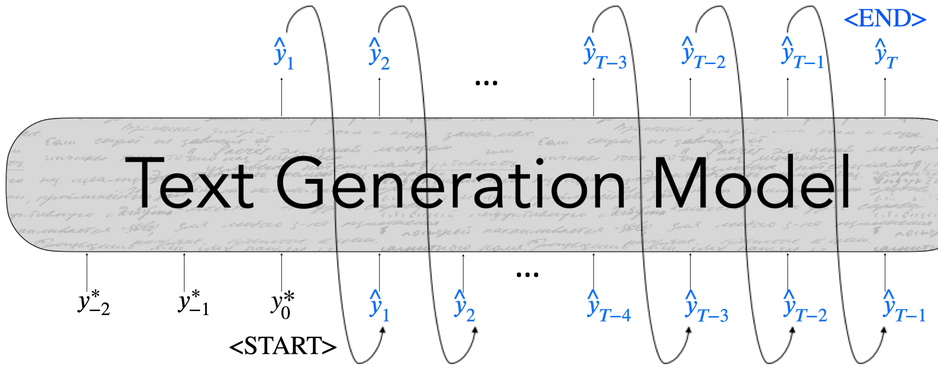


REINFORCE: Basics

- Sample a sequence from your model

$$\mathscr{L}_{RL} = -R(\hat{Y}) \sum_{t=1}^{T} \log P\left(\hat{y}_t \mid \{y^*\}; \{\hat{y}\}_{<t}\right)$$

Often use an aggregate reward for the sequence — every token gets same reward



Reward Estimation

- How should we define a reward function? Just use your evaluation metric!
  - **BLEU** (machine translation; Ranzato et al., ICLR 2016; Wu et al., 2016)
  - **ROUGE** (summarization; Paulus et al., ICLR 2018; Celikyilmaz et al., NAACL 2018)
  - CIDEr (image captioning; Rennie et al., CVPR 2017)
  - SPIDEr (image captioning; Liu et al., ICCV 2017)

Reward Estimation

- How should we define a reward function? Just use your evaluation metric!
  - **BLEU** (machine translation; Ranzato et al., ICLR 2016; Wu et al., 2016)
  - **ROUGE** (summarization; Paulus et al., ICLR 2018; Celikyilmaz et al., NAACL 2018)
  - CIDEr (image captioning; Rennie et al., CVPR 2017)
  - SPIDEr (image captioning; Liu et al., ICCV 2017)

- Be careful about optimizing for the task as opposed to "gaming" the reward!
  - Evaluation metrics are merely proxies for generation quality!
  - "even though RL refinement can achieve better BLEU scores, it barely improves the human impression of the translation quality" – Wu et al., 2016

Reward Estimation

- What behaviors can we tie to rewards?
  - Cross-modality consistency in image captioning (Ren et al., CVPR 2017)
  - Sentence simplicity (Zhang and Lapata, EMNLP 2017)
  - Temporal Consistency (Bosselut et al., NAACL 2018)
  - Utterance Politeness (Tan et al., TACL 2018)
  - Paraphrasing (Li et al., EMNLP 2018)
  - Sentiment (Gong et al., NAACL 2019)
  - Formality (Gong et al., NAACL 2019)

- If you can formalize a behavior as a reward function (or train a neural network to approximate it!), you can train a text generation model to exhibit that behavior!

Implementation Thoughts

- Credit Assignment

$r(\hat{y}_t)$ vs. $R(\hat{Y})$

Implementation Thoughts

- Credit Assignment

$$r(\hat{y}_t) \quad \text{vs.} \quad R(\hat{Y})$$

- Set appropriate baseline

$$\mathscr{L}_{RL} = -\sum (r(\hat{y}_t) - b) \log P\left(\hat{y}_t \mid \{\hat{y}\}_{<t}; \{y^*\}\right)$$

Implementation Thoughts

- Credit Assignment

$$r(\hat{y}_t) \quad \text{vs.} \quad R(\hat{Y})$$

- Set appropriate baseline

$$\mathscr{L}_{RL} = -\sum (r(\hat{y}_t) - b) \log P\left(\hat{y}_t \mid \{\hat{y}\}_{<t}; \{y^*\}\right)$$

- Mix with MLE

$$\mathscr{L} = \mathscr{L}_{MLE} + \alpha \mathscr{L}_{RL}$$

How could I use a “politeness classifier” as a reward function?

How might my model learn to exploit this?

Reward sequences that are classified as polite by the model.

Learn to add please 20x to the end of any utterance.

Training: Takeaways

- ***Teacher forcing*** is still the premier algorithm for training text generation models
- **Diversity** is an issue with sequences generated from teacher forced models
- **Exposure bias** causes text generation models to **lose coherence** easily
  - Models must learn to recover from their own bad samples (e.g., scheduled sampling, RL)
- Training with RL can also allow models to learn behaviors that are challenging to formalize
  - Learning can be very **unstable**!

Decoding References

[1] Gulcehre et al., On Using Monolingual Corpora in Neural Machine Translation. arXiv 2015

[2] Wu et al., Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arxiv 2016

[3] Venugopalan et al., Improving LSTM-based Video Description with Linguistic Knowledge Mined from Text. EMNLP 2016

[4] Li et al., A Diversity-Promoting Objective Function for Neural Conversation Models. EMNLP 2018

[5] Paulus et al., A Deep Reinforced Model for Abstractive Summarization. ICLR 2018

[6] Celikyilmaz et al., Deep Communicating Agents for Abstractive Summarization. NAACL 2018

[7] Holtzman et al., Learning to Write with Cooperative Discriminators. ACL 2018

[8] Fan et al., Hierarchical Neural Story Generation. ACL 2018

[9] Gabriel et al., Discourse Understanding and Factual Consistency in Abstractive Summarization. EACL 2021

[10] Dathathri et al., Plug and Play Language Models: A Simple Approach to Controlled Text Generation. ICLR 2020

[11] Holtzman et al., The Curious Case of Neural Text Degeneration. ICLR 2020

[12] Khandelwal et al., Generalization through Memorization: Nearest Neighbor Language Models. ICLR 2020

Training References

[1] Bengio et. al., Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. NeurIPS 2015

[2] Ranzato et al., Sequence Level Training with Recurrent Neural Networks. ICLR 2016

[3] Wu et al., Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. 2016

[4] Ren et al., Deep Reinforcement Learning-based Image Captioning with Embedding Reward. CVPR 2017

[5] Rennie et al., Self-critical Sequence Training for Image Captioning. CVPR 2017

[6] Liu et al., Improved Image Captioning via Policy Gradient Optimization of SPIDEr. ICCV 2017

[7] Zhang and Lapata, Sentence Simplification with Deep Reinforcement Learning. EMNLP 2017

[8] Paulus et al., A Deep Reinforced Model for Abstractive Summarization. ICLR 2018

[9] Celikyilmaz et al., Deep Communicating Agents for Abstractive Summarization. NAACL 2018

[10] Bosselut et al., Discourse-Aware Neural Rewards for Coherent Text Generation. NAACL 2018

[11] Tan and Bansal, Polite Dialogue Generation Without Parallel Data. TACL 2018

[12] Li et al., Paraphrase Generation with Deep Reinforcement Learning. EMNLP 2018

[13] Holtzman et. al., The Curious Case of Neural Text Degeneration. ICLR 2020