Lecture 17:
Generating From Language Models II

Instructor: Swabha Swayamdipta
USC CSCI 499 LMs in NLP
Apr 1, Spring 2024

Slides adapted from Chris Manning, Xiang Lisa Li

Logistics / Announcements

Logistics / Announcements

- Today: Quiz 5
- Wednesday: HW4 due
- HW3 Grades: Out latest by tomorrow

- Upcoming Guest Lectures
  - Lecture on Prompting: Qinyuan Ye
  - Lecture on Alignment: Justin Cho

Lecture Outline

- Mid-Semester Feedback
- Recap: Tokenization
- Recap: Natural Language Generation - Basics
- Recap: Classic Inference Algorithms: Greedy, Exhaustive and Beam Search
- Modern Generation Algorithms
- Evaluating Generations
- Quiz 5

Recap: Tokenization in Transformers

Byte-pair encoding

- Byte-pair encoding is a simple, effective strategy for defining a subword vocabulary
- Adapted for word segmentation from data compression technique (Gage, 1994)
  - Instead of merging frequent pairs of bytes, we merge characters or character sequences
- Algorithm:
  1. Start with a vocabulary containing only characters and an "end-of-word" symbol.
  2. Using a corpus of text, find *the most common adjacent characters* "a,b"; add "ab" as a subword
     - This is a learned operation!
     - Only combine pairs (hence the name!)
  3. Replace instances of the character pair with the new subword; repeat until desired vocabulary size.
- At test time, first split words into sequences of characters, then apply the learned operations to merge the characters into larger, known symbols
- Originally used in NLP for machine translation; now a similar method (WordPiece) is used in pretrained models.

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t</w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t</w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>

Vocabulary

d	e	i	l	n	o	s	t	w

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t</w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>

Vocabulary

d	e	i	l	n	o	s	t	w

Frequency

d-e (3)	l-o (7)	t-</w> (8)
e-r (2)	n-e (5)	w-</w> (5)
e-s (8)	o-w (7)	w-e (7)
e-w (5)	r-</w> (2)	w-i (3)
i-d (3)	s-t (8)	

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t</w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es								

Frequency

d-e (3)	l-o (7)	t-</w> (8)
e-r (2)	n-e (5)	w-</w> (5)
e-s (8)	o-w (7)	w-e (7)
e-w (5)	r-</w> (2)	w-i (3)
i-d (3)	s-t (8)	

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w es t </w>
l o w </w>	l o w e r </w>	n e w es t </w>
l o w </w>	w i d es t </w>	n e w es t </w>
l o w </w>	w i d es t</w>	n e w es t </w>
l o w </w>	w i d es t </w>	n e w es t </w>

Frequency

d-es (3)	l-o (7)	w-</w> (5)
e-r (2)	n-e (5)	w-es (5)
e-w (5)	o-w (7)	w-e (2)
es-t (8)	r-</w> (2)	w-i (3)
i-d (3)	t-</w> (8)	

Vocabulary

d	e	i	l	n	o	s	t	w
es								

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

USC Viterbi

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w es t </w>
l o w </w>	l o w e r </w>	n e w es t </w>
l o w </w>	w i d es t </w>	n e w es t </w>
l o w </w>	w i d es t</w>	n e w es t </w>
l o w </w>	w i d es t </w>	n e w es t </w>

Frequency

d-es (3)	l-o (7)	w-</w> (5)
e-r (2)	n-e (5)	w-es (5)
e-w (5)	o-w (7)	w-e (2)
es-t (8)	r-</w> (2)	w-i (3)
i-d (3)	t-</w> (8)	

Vocabulary

d	e	i	l	n	o	s	t	w
es	est							

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

USC Viterbi

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w est </w>
l o w </w>	l o w e r </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es	est							

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

USC Viterbi

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w est </w>
l o w </w>	l o w e r </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es	est	est</w>	lo	low	low</w>	ne	new	newest</w>

After 10 merges

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

Recap: Natural Language Generation

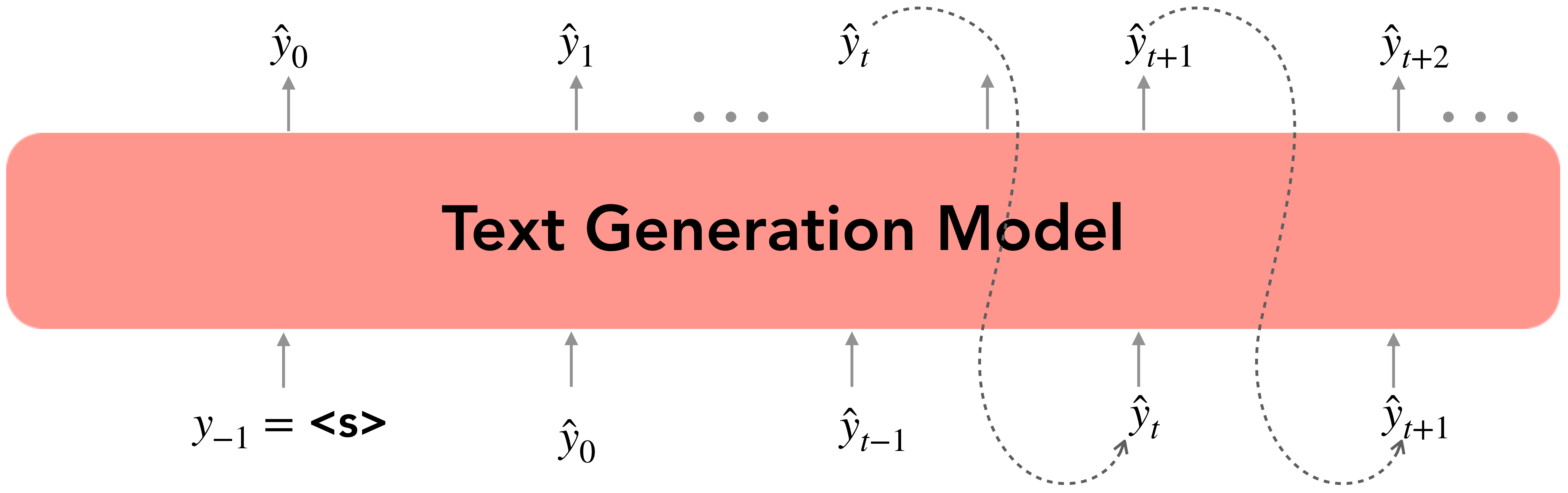
USC Viterbi

Language Generation: Fundamentals

In autoregressive text generation models, at each time step t, our model takes in a sequence of tokens as input $S = f_\theta(y_{<t}) \in \mathbb{R}^V$ and outputs a new token, $\hat{y}_t$

For model $f_\theta(\cdot)$ and vocabulary V, we get scores $S = f_\theta(y_{<t}) \in \mathbb{R}^V$

$$P(w \mid y_{<t}) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

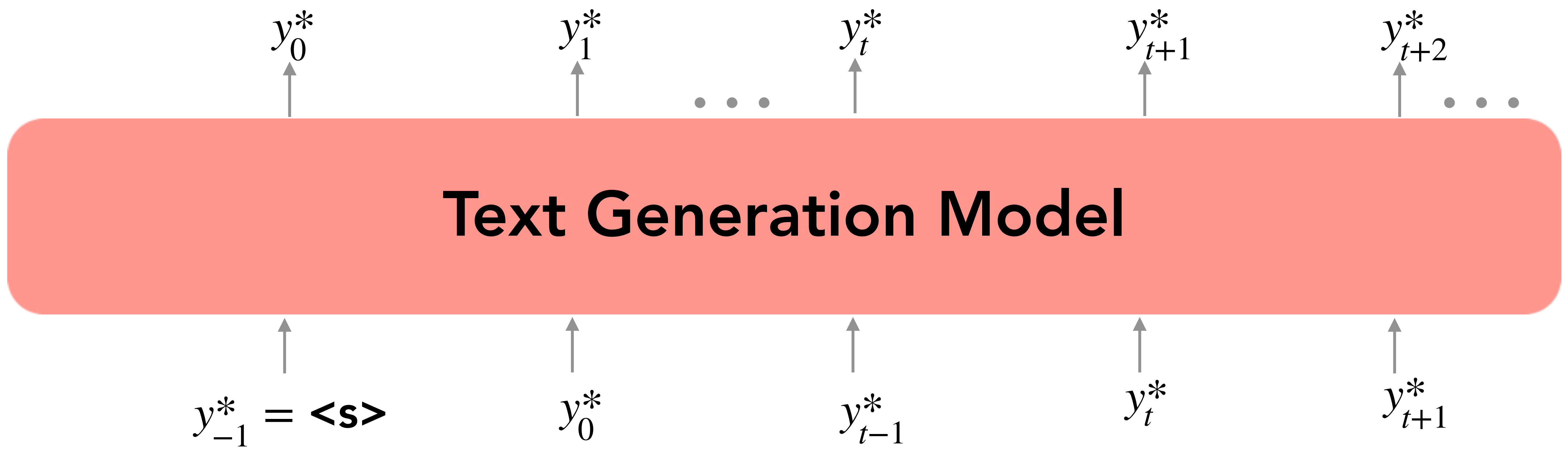


Language Generation: Training

- Trained one token at a time to maximize the probability of the next token y_t^* given preceding words $y_{<t}^*$

$$\mathscr{L} = -\sum_{t=1}^{T} \log P(y_t \mid y_{<t}) = -\sum_{t=1}^{T} \log \frac{\exp(S_{y_t \mid y_{<t}})}{\sum_{v \in V} \exp(S_{v \mid y_{<t}})}$$

- Classification task at each time step trying to predict the actual word y_t^* in the training data
- "Teacher forcing" (reset at each time step to the ground truth)



Language Generation: Inference

- At inference time, our decoding algorithm defines a function to select a token from this distribution:

Inference / Decoding Algorithm

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

- The "obvious" decoding algorithm is to greedily choose the highest probability next token according to the model at each time step

$$g = \arg\max$$

$$\hat{y}_t = \arg\max_{w \in V}(P(y_t = w \mid y_{<t}))$$

Recap: Classic Inference Algorithms: Greedy and Beam Search

Decoding

- Generation from a language model is also called decoding / inference
- Strategy so far is **Greedy**: Take $\arg\max$ on each step of the decoder to produce the most probable word on each step
    - Not looking ahead, making the hastiest decision given all the information we have
    - Problem: No wiggle room for errors
    - Problem: Bland / repetitive generations (degeneracy)

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...**

Holtzmann et al., 2020

Exhaustive Search Decoding

- Ideally, we want to find a (length T) translation y that maximizes

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x)$$
$$= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)$$

- We could try computing all possible sequences y
  - This means that on each step t of the decoder, we're tracking V^t possible partial translations, where V is the vocabulary size
  - This $O(V^T)$complexity is far too expensive!

Beam Search Decoding

Beam Search Decoding

- Core idea: On each step of decoder, keep track of the k most probable partial translations (which we call hypotheses)
  - k is the beam size (in practice around 5 to 10, in NMT)

Beam Search Decoding

- Core idea: On each step of decoder, keep track of the k most probable partial translations (which we call hypotheses)
  - k is the beam size (in practice around 5 to 10, in NMT)
- A hypothesis has a score which is its log probability:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses, tracking top k on each step

Beam Search Decoding

- Core idea: On each step of decoder, keep track of the k most probable partial translations (which we call hypotheses)
  - k is the beam size (in practice around 5 to 10, in NMT)
- A hypothesis has a score which is its log probability:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses, tracking top k on each step
- Beam search is not guaranteed to find optimal solution

Beam Search Decoding

- Core idea: On each step of decoder, keep track of the k most probable partial translations (which we call hypotheses)
  - k is the beam size (in practice around 5 to 10, in NMT)
- A hypothesis has a score which is its log probability:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses, tracking top k on each step
- Beam search is not guaranteed to find optimal solution
- But much more efficient than exhaustive search!

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

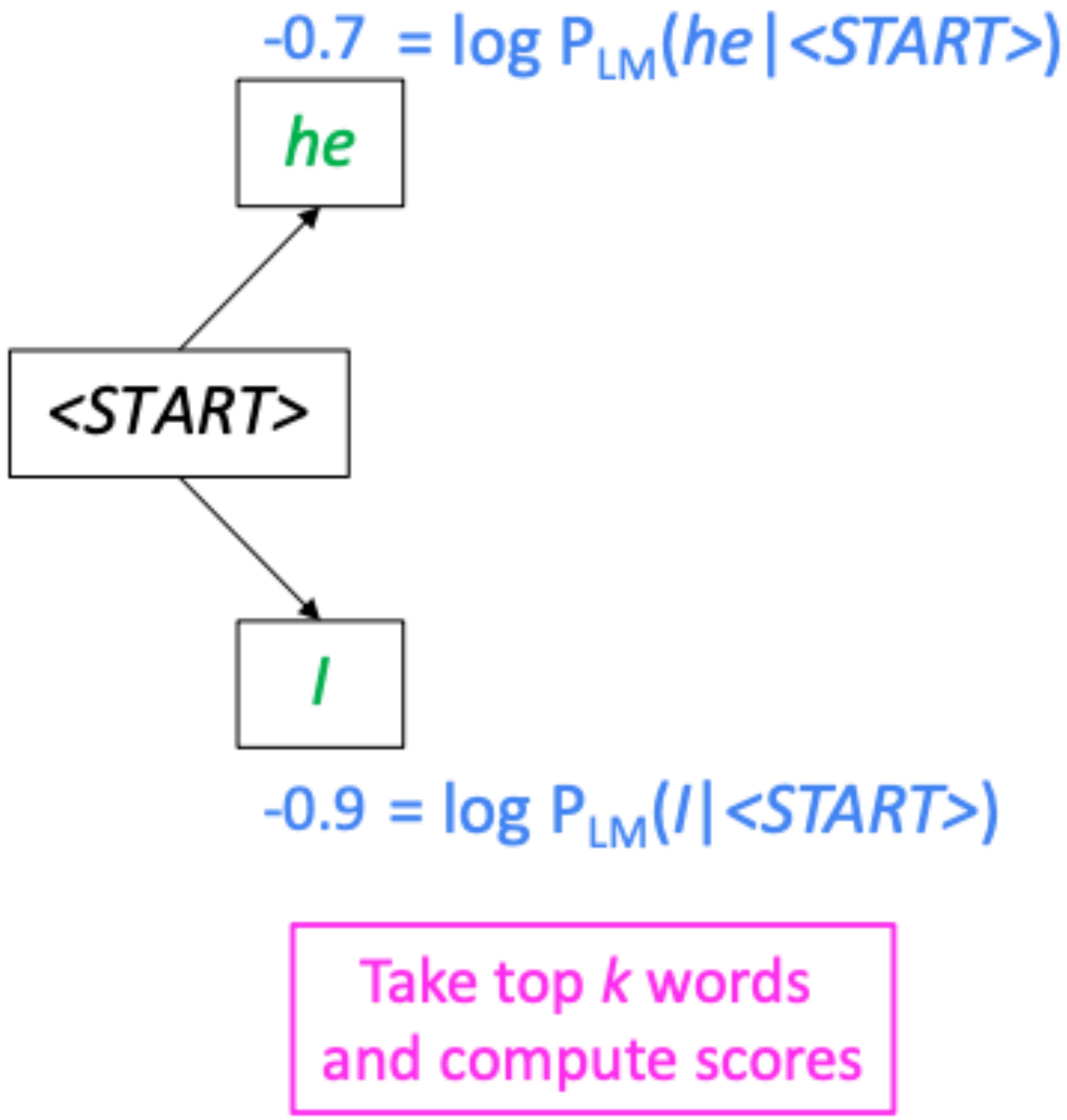
<START>

Calculate prob
dist of next word

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

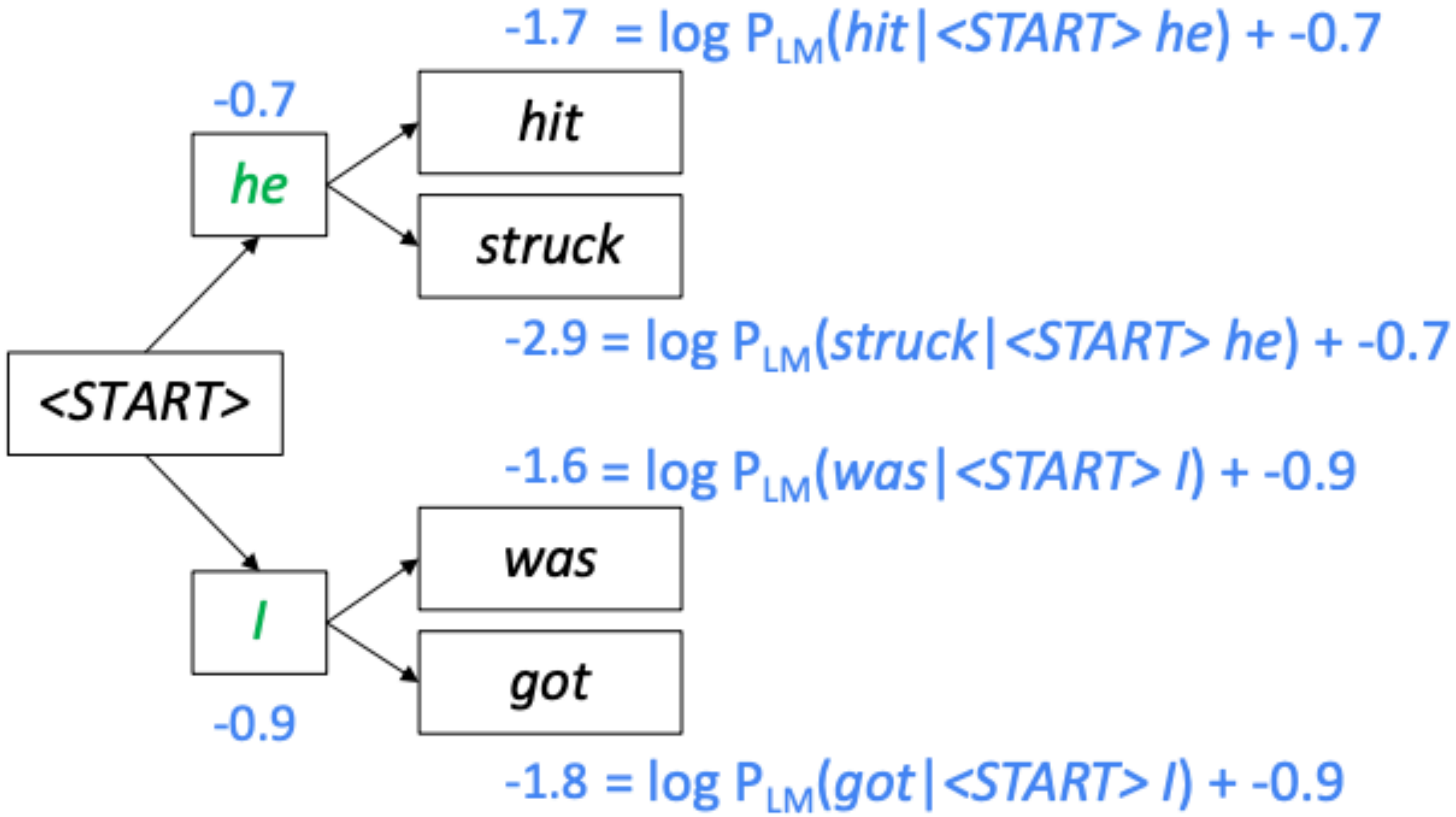


Slide credit: Chris Manning

USC Viterbi

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

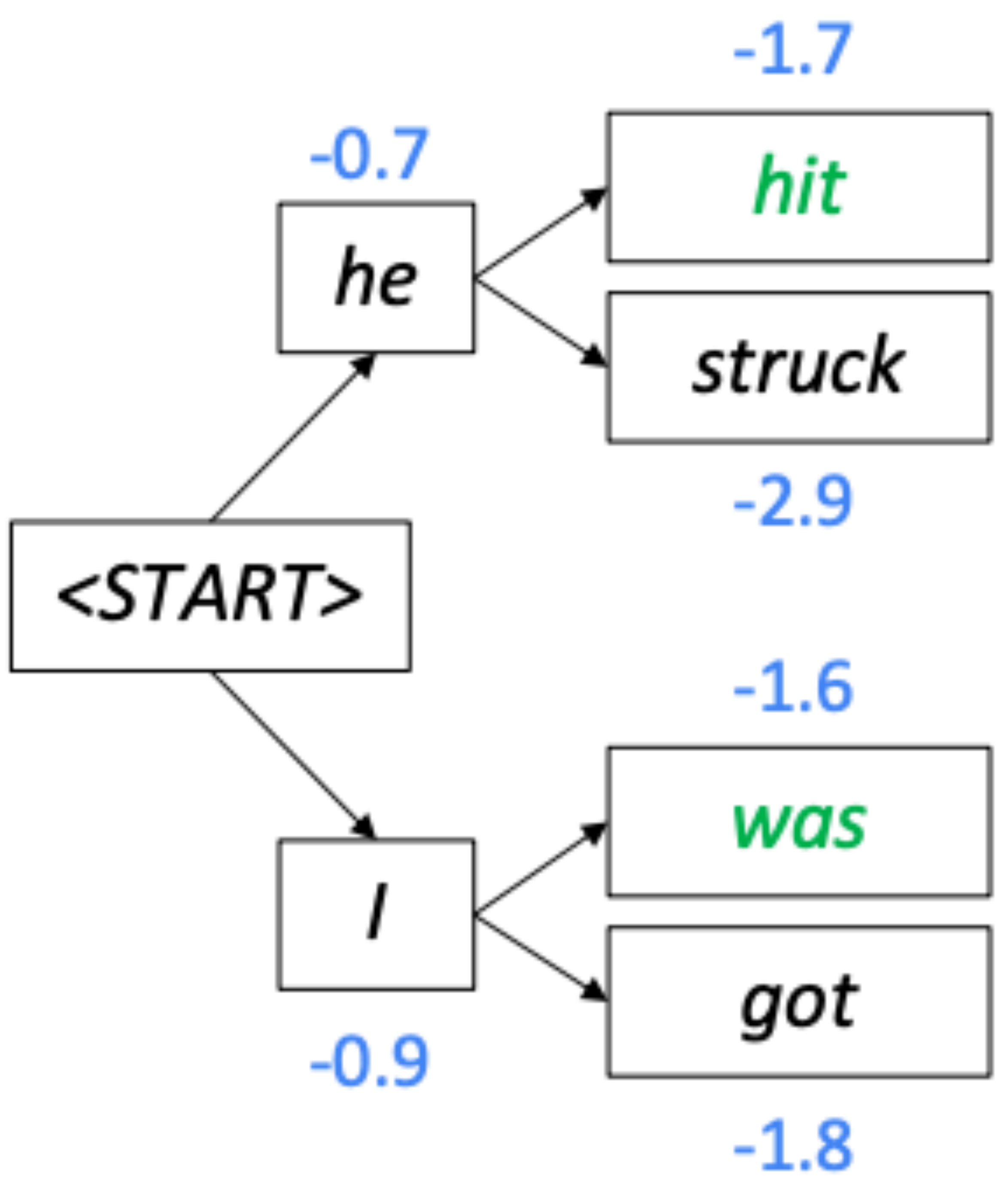


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

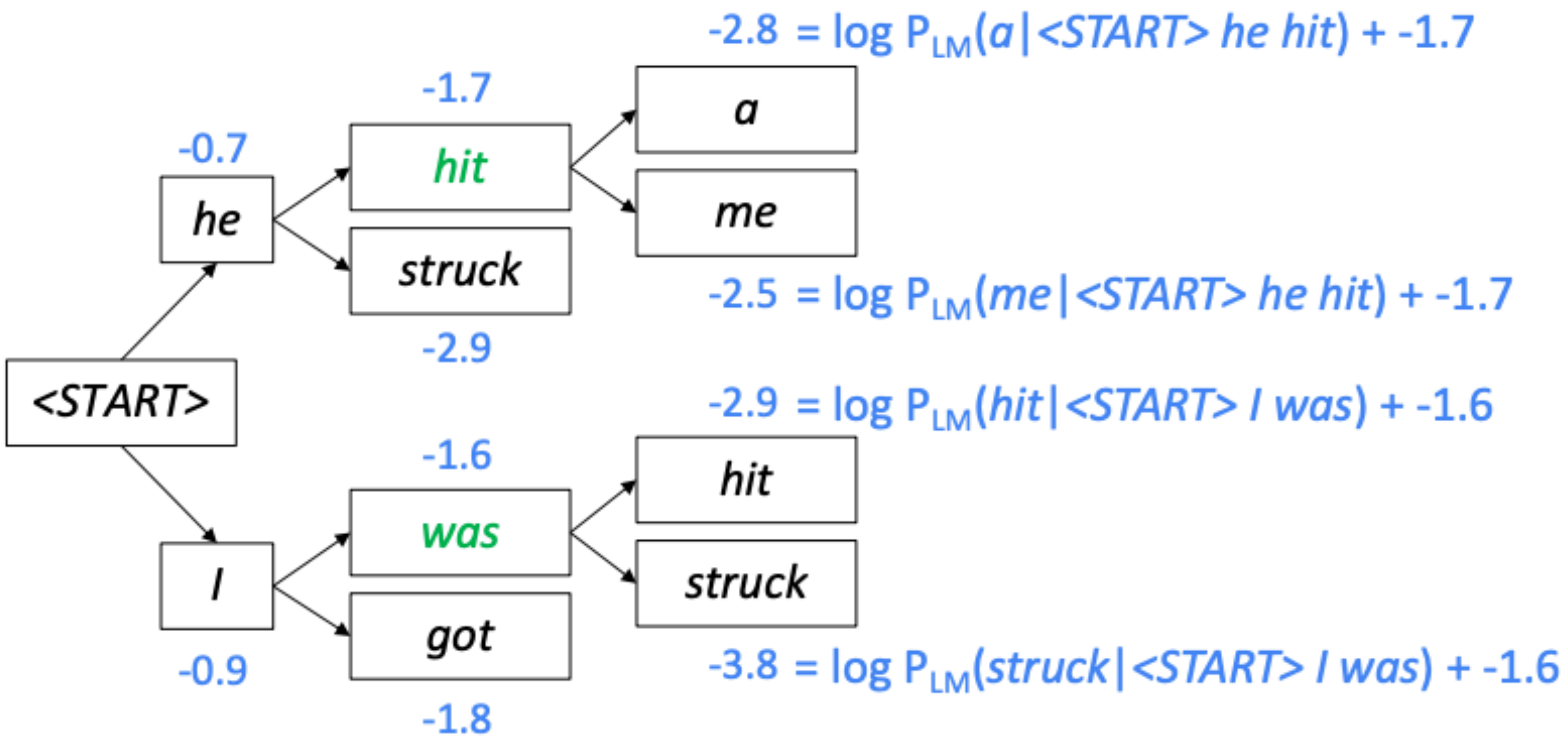


Of these k^2 hypotheses,
just keep *k* with highest scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

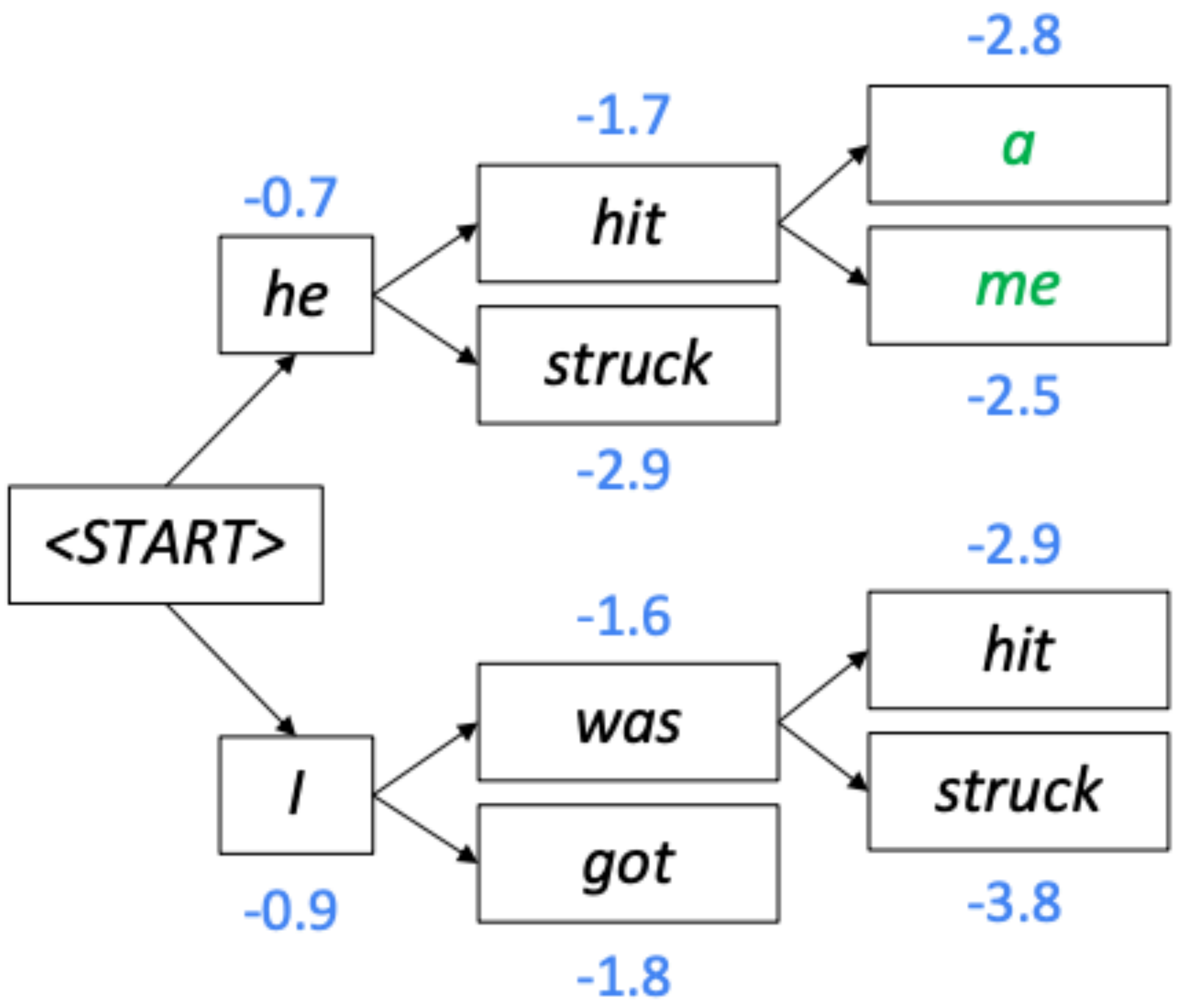


For each of the *k* hypotheses, find top *k* next words and calculate scores

de credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

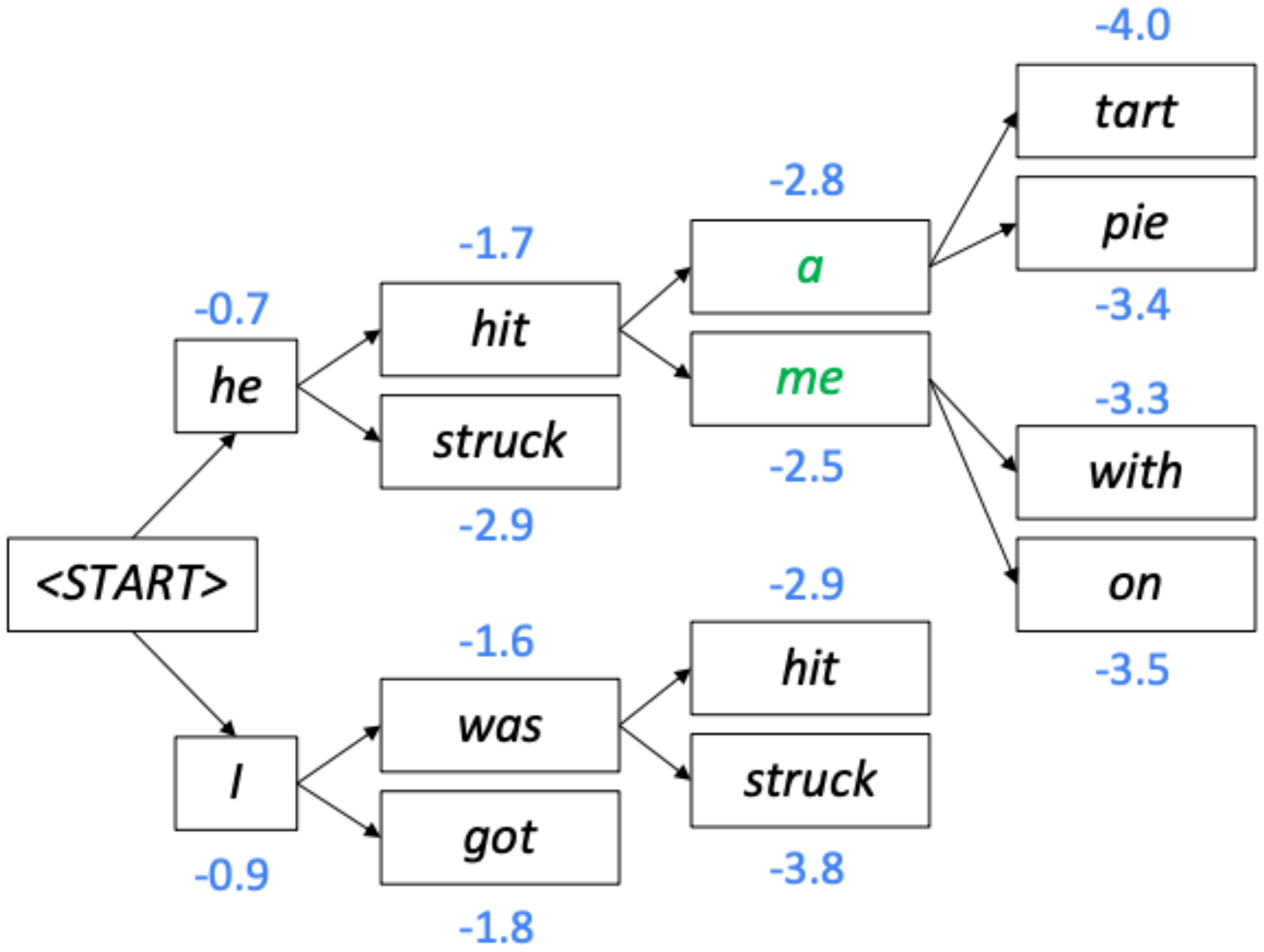


Of these k^2 hypotheses, just keep k with highest scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1,\ldots,y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i|y_1,\ldots,y_{i-1},x)$

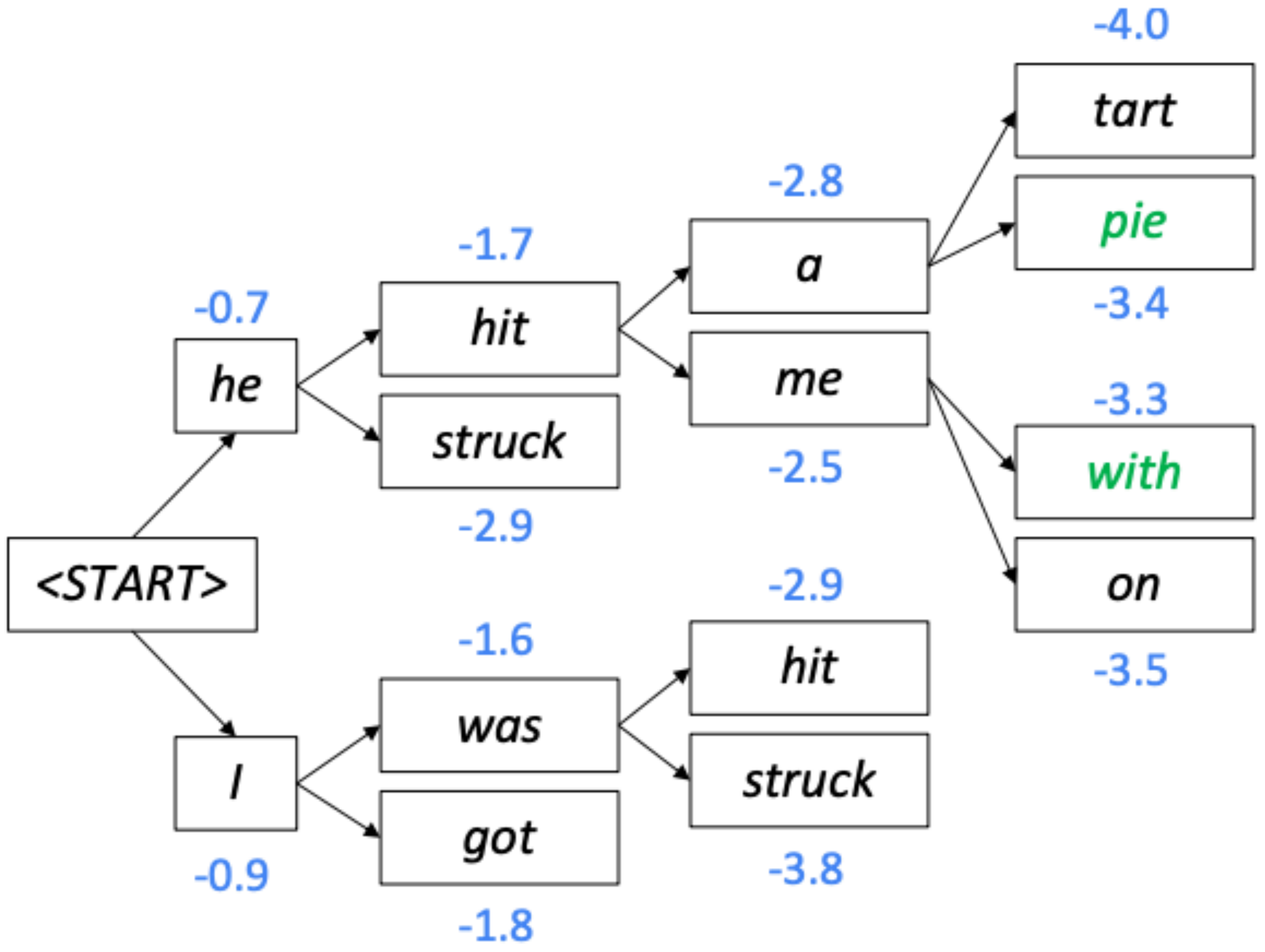


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

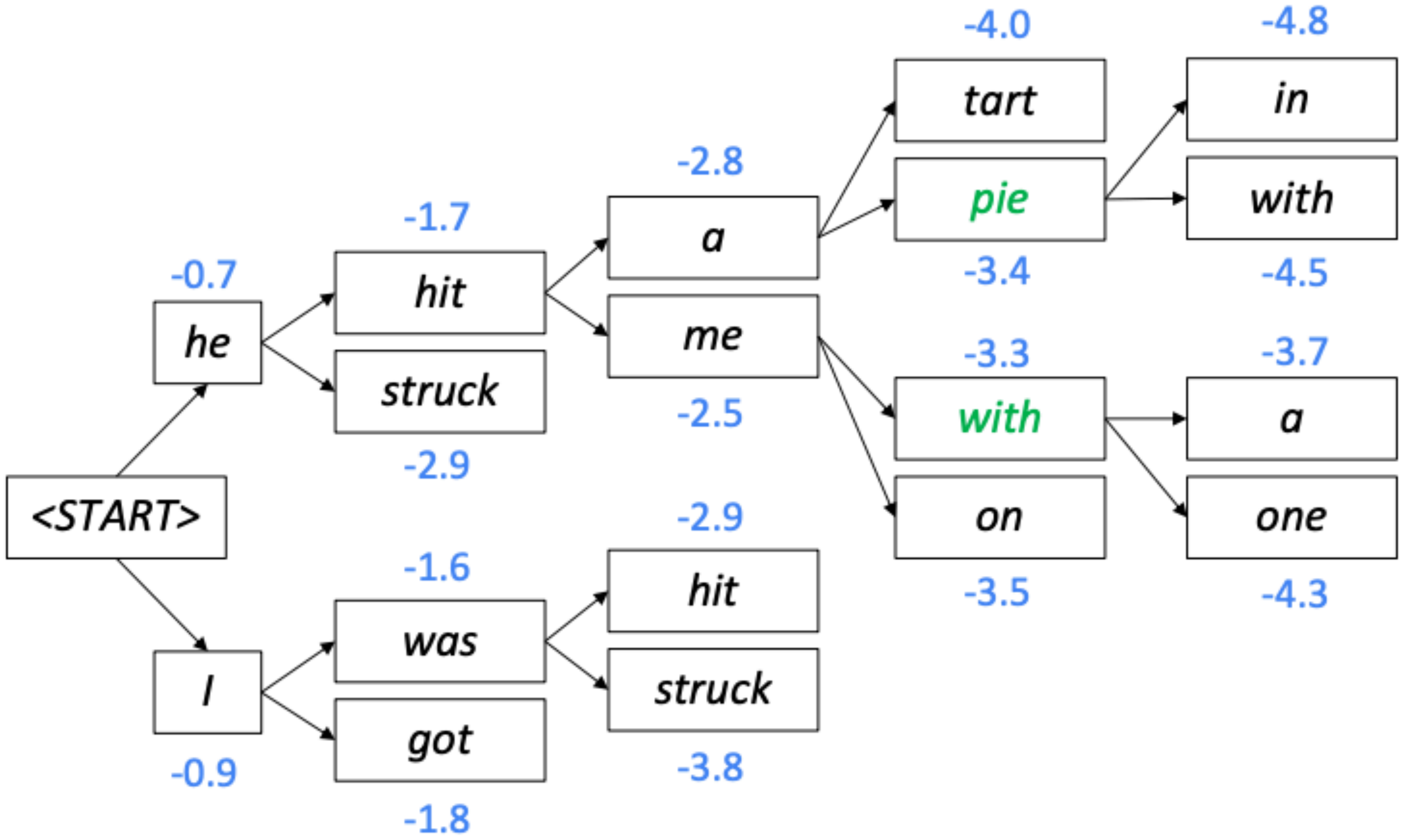


Of these k^2 hypotheses, just keep k with highest scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



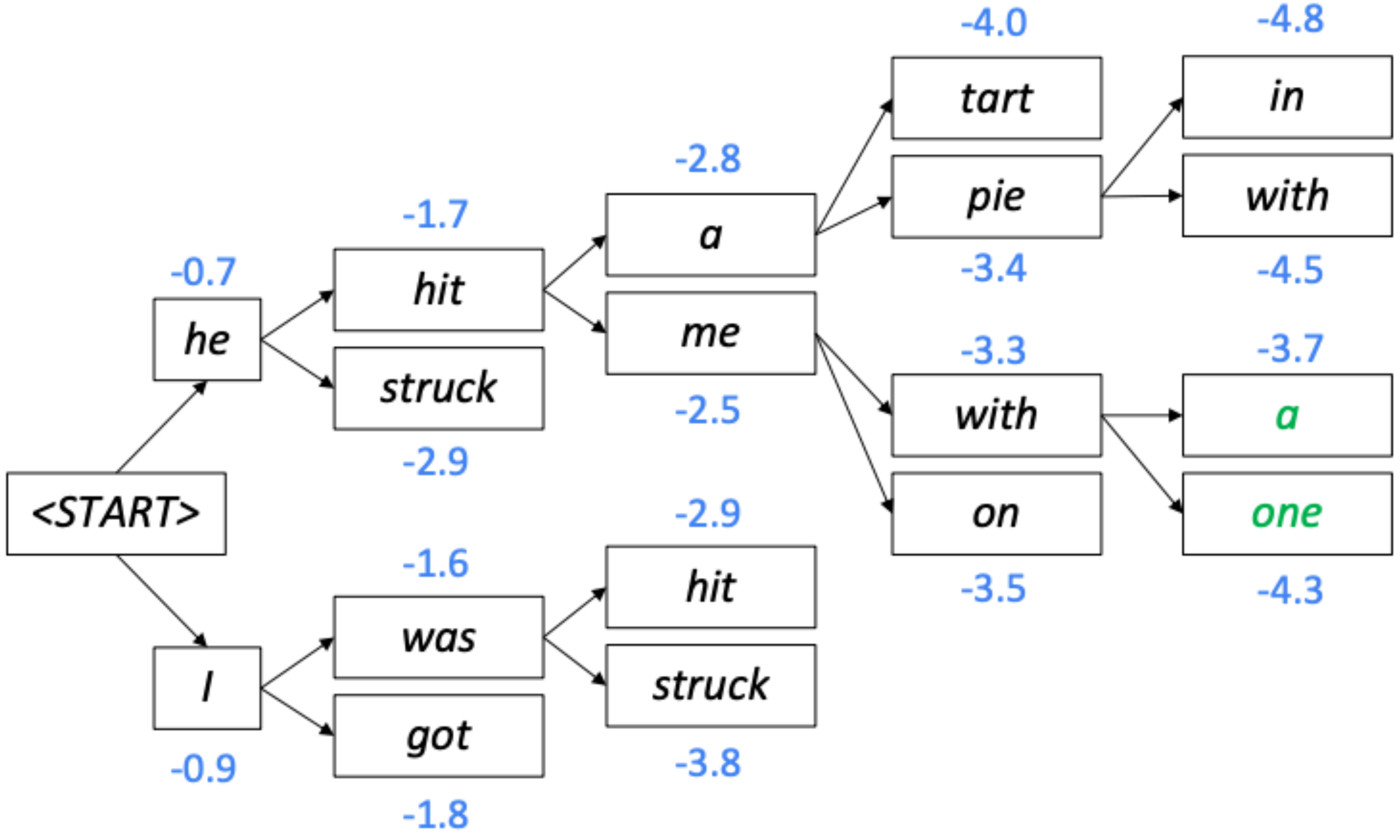
For each of the *k* hypotheses, find top *k* next words and calculate scores

Slide credit: Chris Manning

USC Viterbi

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



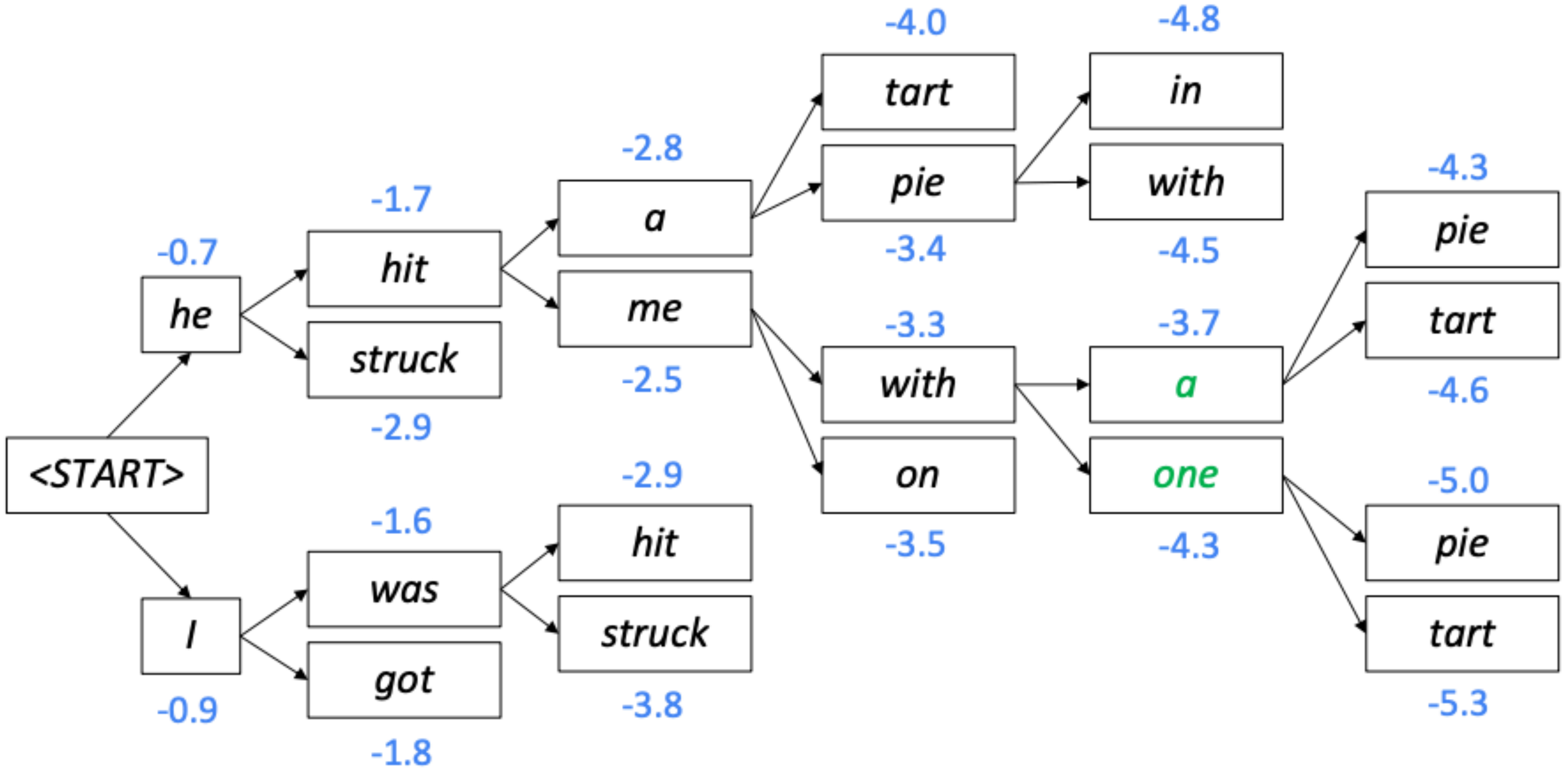
Of these k^2 hypotheses, just keep k with highest scores

Slide credit: Chris Manning

USC Viterbi

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

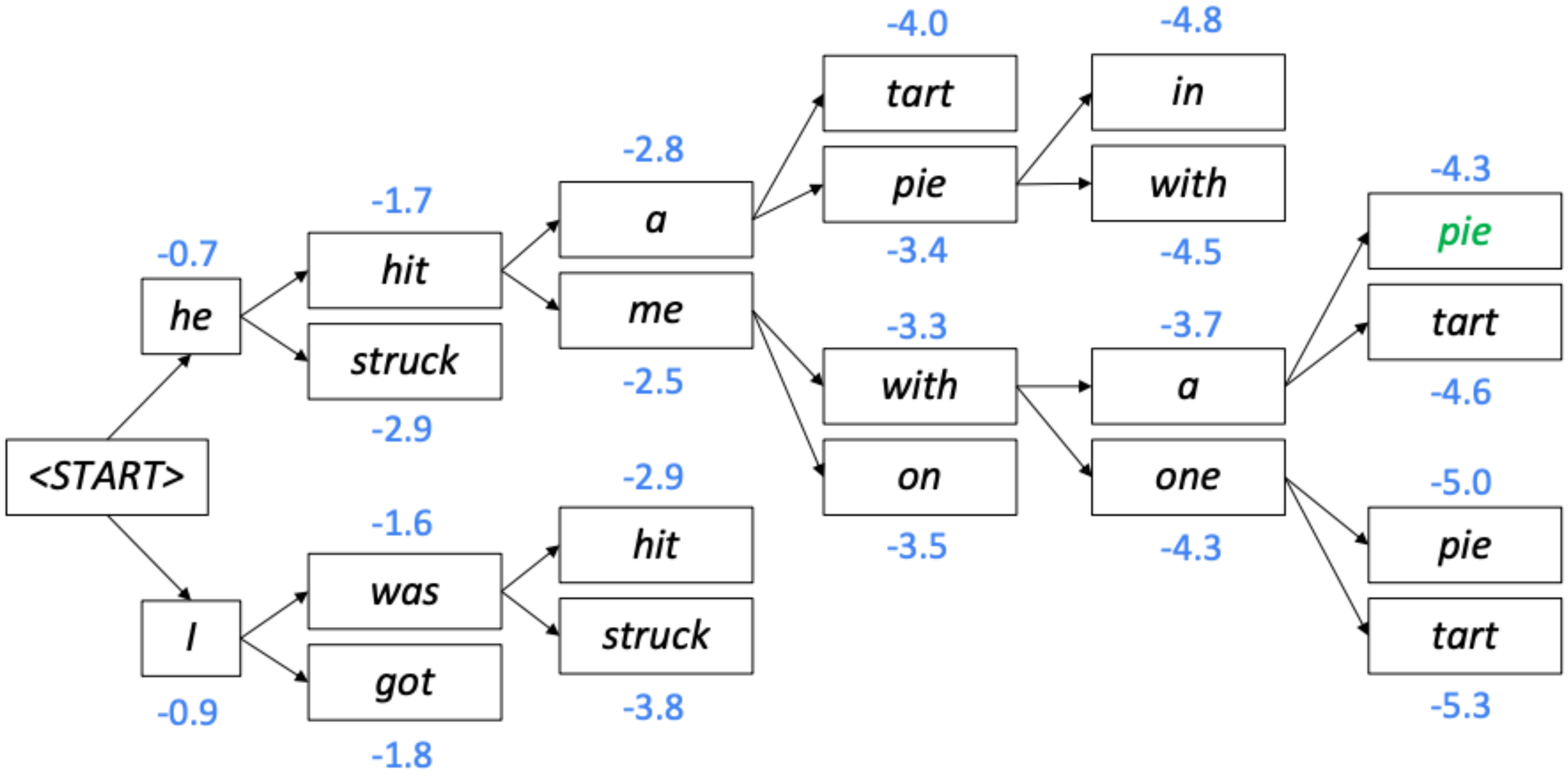


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

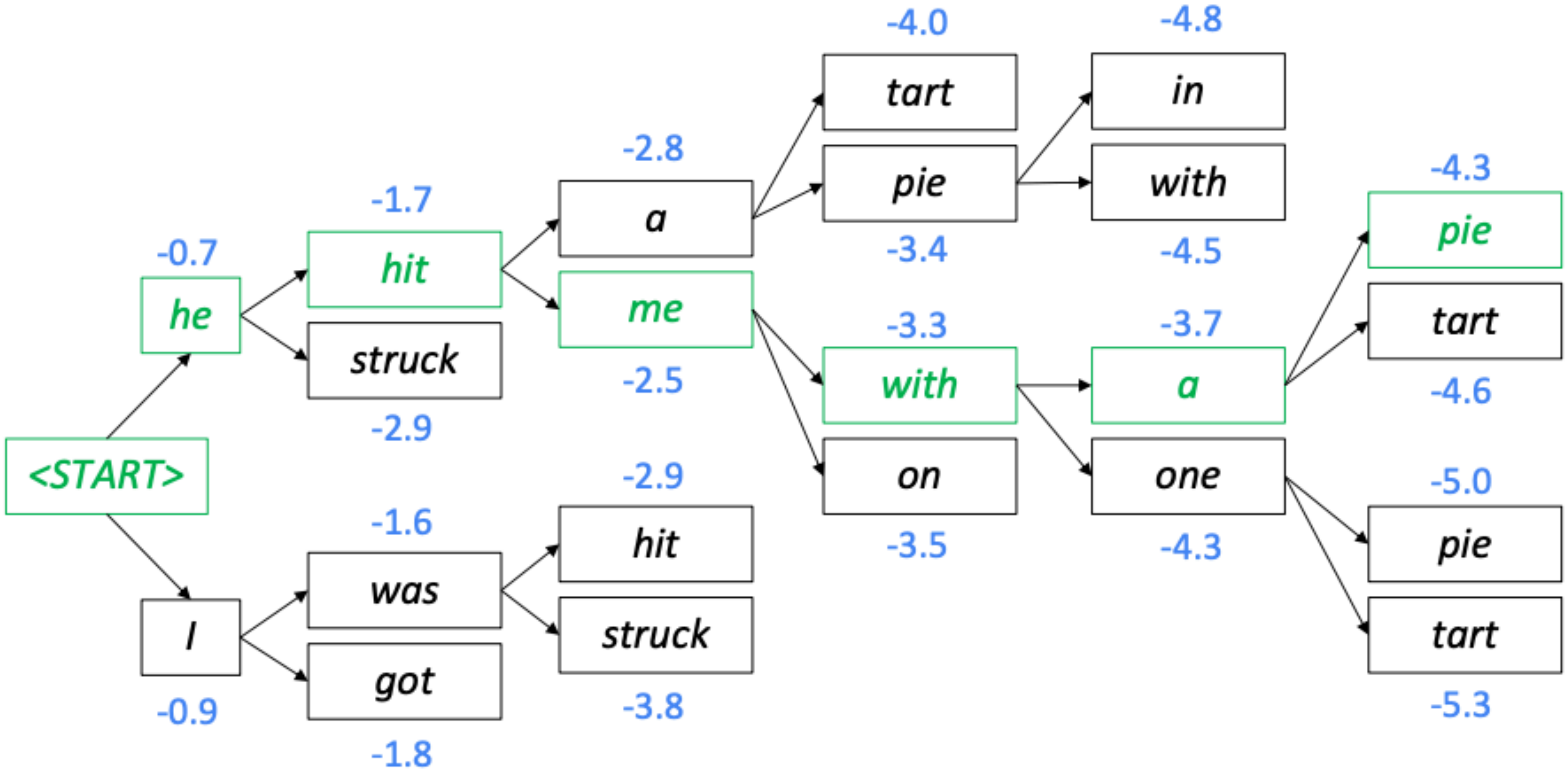


This is the top-scoring hypothesis!

Slide credit: Chris Manning

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Backtrack to obtain the full hypothesis

Slide credit: Chris Manning

USC Viterbi

Beam Search Decoding: Stopping Criterion

Beam Search Decoding: Stopping Criterion

- Greedy Decoding is done until the model produces an </s> token
  - For e.g. <s> he hit me with a pie </s>

Beam Search Decoding: Stopping Criterion

- Greedy Decoding is done until the model produces an </s> token
  - For e.g. <s> he hit me with a pie </s>
- In Beam Search Decoding, different hypotheses may produce </s> tokens at different time steps
  - When a hypothesis produces </s>, that hypothesis is complete.
  - Place it aside and continue exploring other hypotheses via beam search.

Beam Search Decoding: Stopping Criterion

- Greedy Decoding is done until the model produces an </s> token
    - For e.g. <s> he hit me with a pie </s>
- In Beam Search Decoding, different hypotheses may produce </s> tokens at different time steps
    - When a hypothesis produces </s>, that hypothesis is complete.
    - Place it aside and continue exploring other hypotheses via beam search.
- Usually we continue beam search until:
    - We reach time step T (where T is some pre-defined cutoff), or
    - We have at least n completed hypotheses (where n is pre-defined cutoff)

Beam Search Decoding: Stopping Criterion

- Greedy Decoding is done until the model produces an </s> token
    - For e.g. <s> he hit me with a pie </s>
- In Beam Search Decoding, different hypotheses may produce </s> tokens at different time steps
    - When a hypothesis produces </s>, that hypothesis is complete.
    - Place it aside and continue exploring other hypotheses via beam search.
- Usually we continue beam search until:
    - We reach time step T (where T is some pre-defined cutoff), or
    - We have at least n completed hypotheses (where n is pre-defined cutoff)
- Beam Search: Deprioritize short sequences by length normalization

Beam Search Decoding: Parting Thoughts

Beam Search Decoding: Parting Thoughts

- We have our list of completed hypotheses. Now how to select top one?

Beam Search Decoding: Parting Thoughts

- We have our list of completed hypotheses. Now how to select top one?
- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

USC Viterbi

Beam Search Decoding: Parting Thoughts

- We have our list of completed hypotheses. Now how to select top one?
- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

USC Viterbi

Beam Search Decoding: Parting Thoughts

- We have our list of completed hypotheses. Now how to select top one?
- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

- Problem with this: longer hypotheses have lower score

Beam Search Decoding: Parting Thoughts

- We have our list of completed hypotheses. Now how to select top one?
- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t|x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i|y_1, \ldots, y_{i-1}, x)$$

- Problem with this: longer hypotheses have lower score
- Fix: Normalize by length. Use this to select top one instead

Beam Search Decoding: Parting Thoughts

- We have our list of completed hypotheses. Now how to select top one?
- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

- Problem with this: longer hypotheses have lower score
- Fix: Normalize by length. Use this to select top one instead

$$\frac{1}{t} \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

USC Viterbi

Beam Search Decoding: Parting Thoughts

- We have our list of completed hypotheses. Now how to select top one?
- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t|x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i|y_1, \ldots, y_{i-1}, x)$$

- Problem with this: longer hypotheses have lower score
- Fix: Normalize by length. Use this to select top one instead

$$\frac{1}{t}\sum_{i=1}^{t} \log P_{\text{LM}}(y_i|y_1, \ldots, y_{i-1}, x)$$

But this is expensive!

Maximization Based Decoding

Maximization Based Decoding

- Either greedy or beam search

Maximization Based Decoding

- Either greedy or beam search
- Beam search can be more effective with large beam width, but also more expensive

Maximization Based Decoding

- Either greedy or beam search
- Beam search can be more effective with large beam width, but also more expensive
- Another key issue:

Maximization Based Decoding

- Either greedy or beam search
- Beam search can be more effective with large beam width, but also more expensive
- Another key issue:

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...**

Holtzmann et al., 2020

USC Viterbi

Maximization Based Decoding

- Either greedy or beam search
- Beam search can be more effective with large beam width, but also more expensive
- Another key issue:

Generation can be bland or repetitive (also called degenerate)

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...**

Holtzmann et al., 2020

Maximization Based Decoding

- Either greedy or beam search
- Beam search can be more effective with large beam width, but also more expensive
- Another key issue:

Generation can be bland or repetitive (also called degenerate)

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.

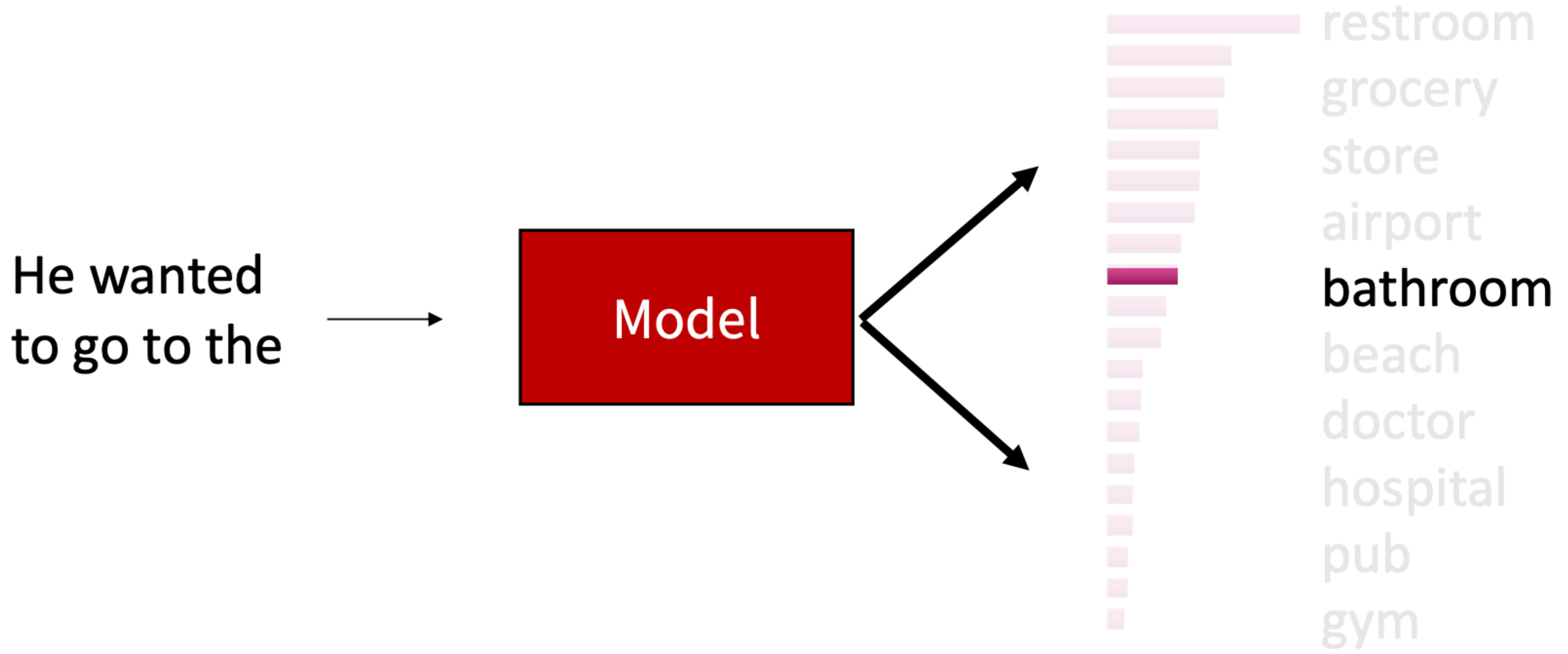
Continuation: The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the **Universidad Nacional Autónoma de México (UNAM)** and **the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma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

Perhaps we should not really be maximizing!
What else could we do?

Holtzmann et al., 2020

Solution: Don't Maximize, Pick a Sample

- Sample a token from the distribution of tokens.
- But this is not a random sample, it is a sample for the learned model distribution
  - Respects the probabilities, without going just for the maximum probability option
  - Or else, you would get something meaningless
  - Many good options which are not the maximum probability!

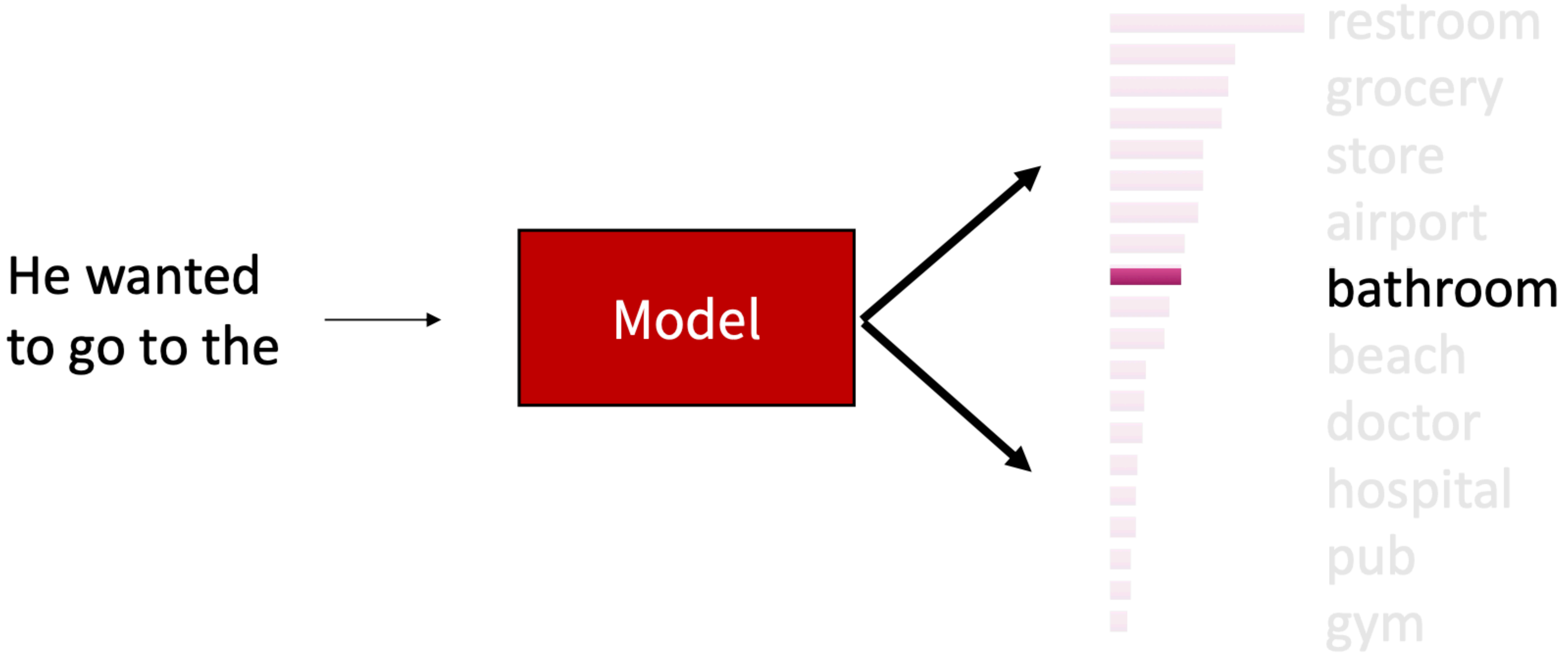


Modern Generation: Sampling

USC Viterbi

Pure / Ancestral Sampling

$$y_t \sim P_t(w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

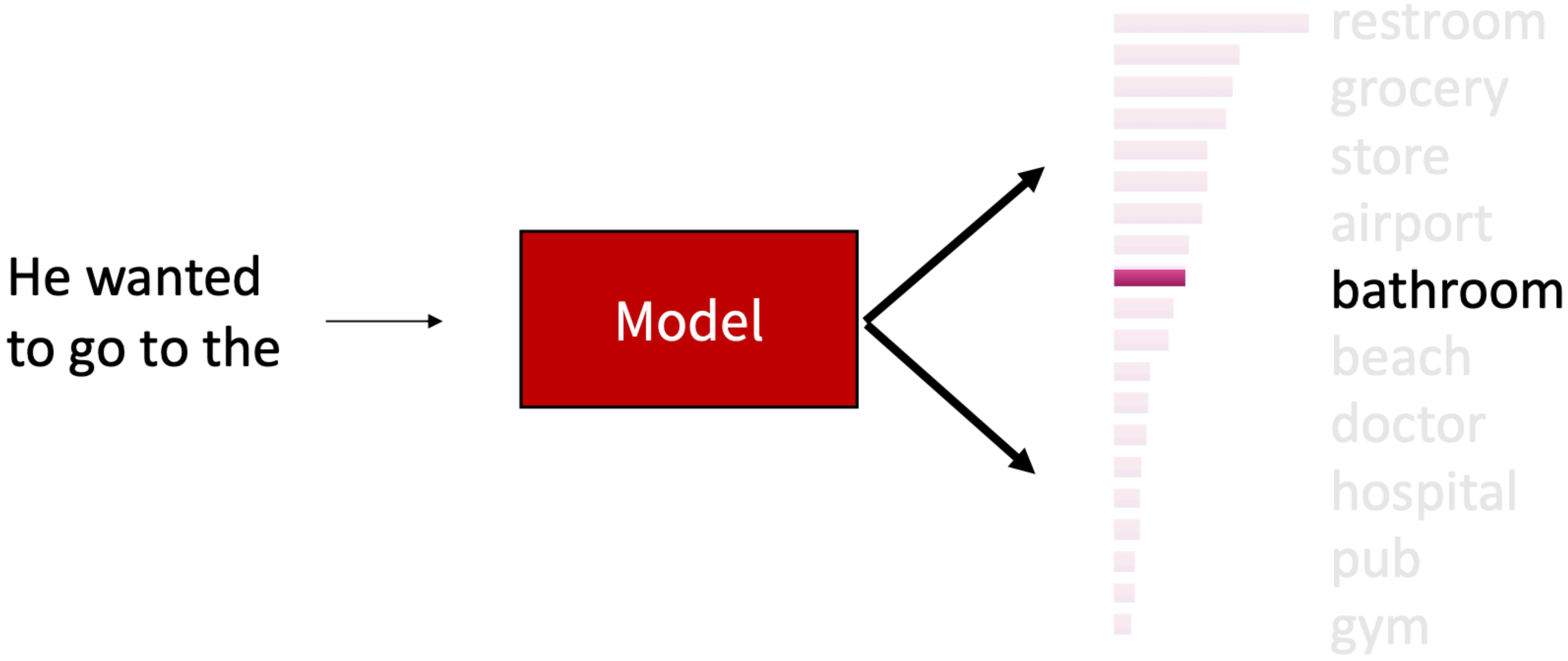


USC Viterbi

Pure / Ancestral Sampling

$$y_t \sim P_t(w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

- Sample directly from P_t

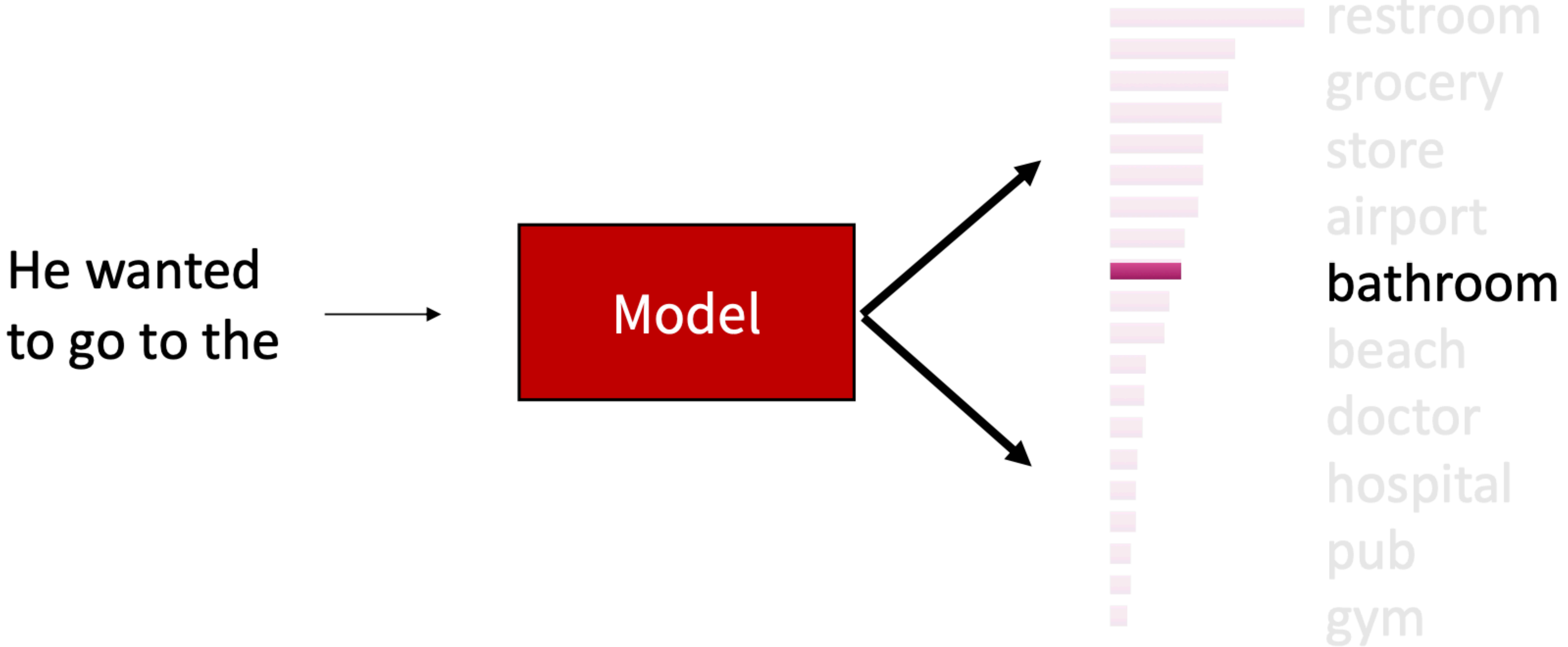


USC Viterbi

Pure / Ancestral Sampling

$$y_t \sim P_t(w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

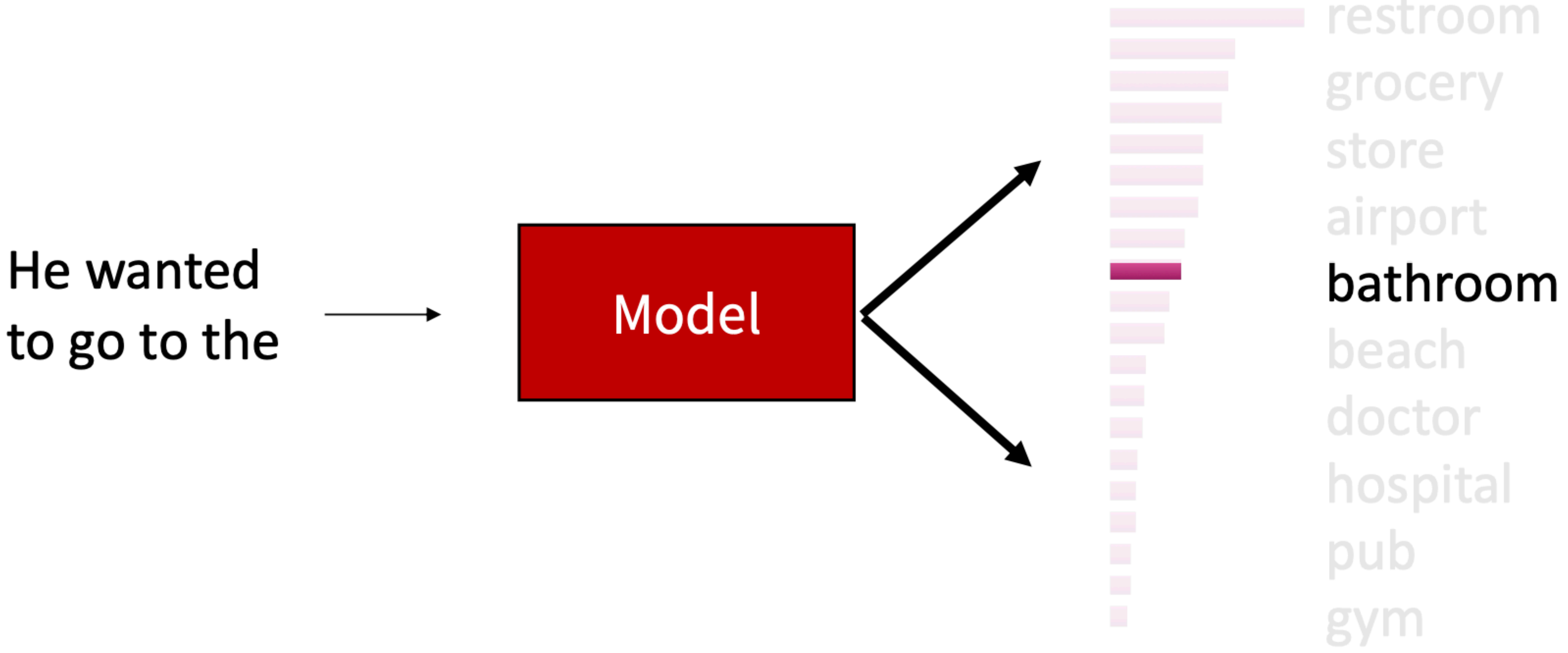
- Sample directly from P_t
- Still has access to the entire vocabulary



Pure / Ancestral Sampling

$$y_t \sim P_t(w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

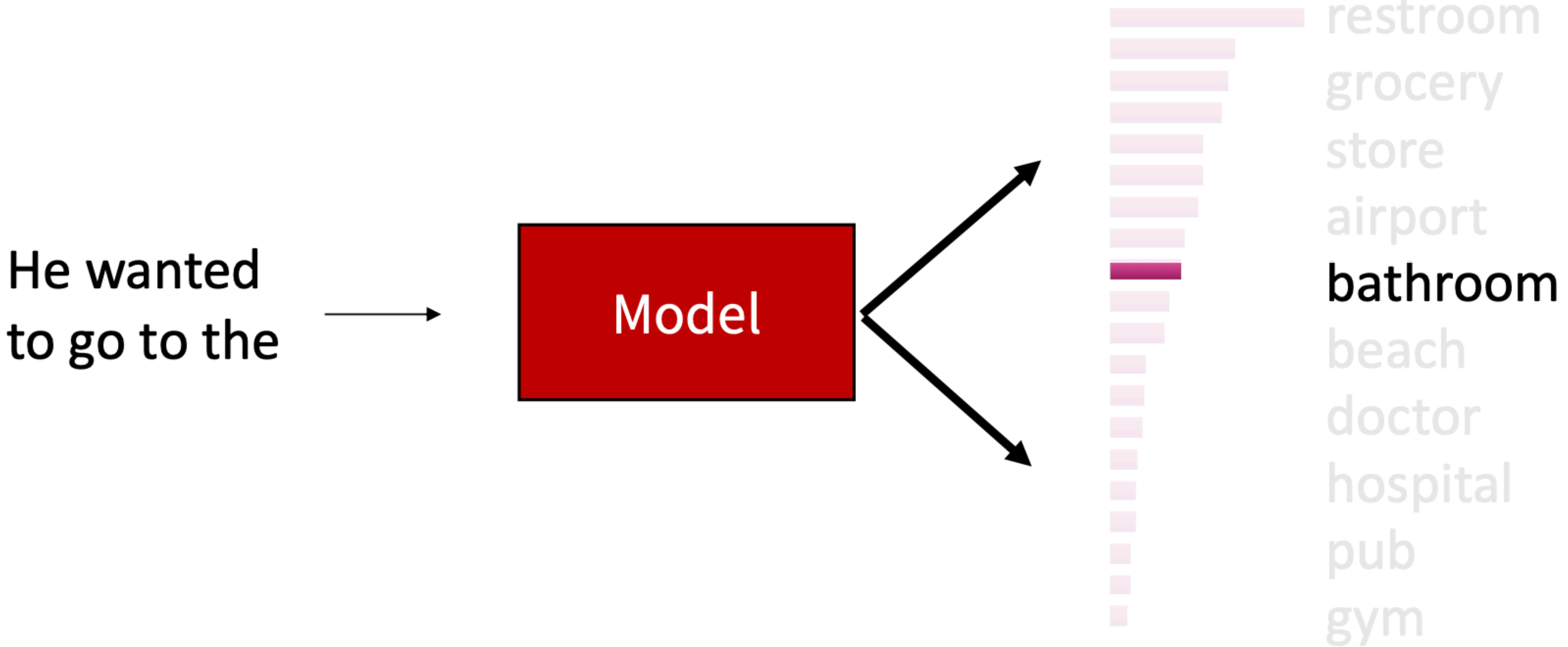
- Sample directly from P_t
- Still has access to the entire vocabulary
- But if the model distributions are of low quality, generations will be of low quality as well



Pure / Ancestral Sampling

$$y_t \sim P_t(w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

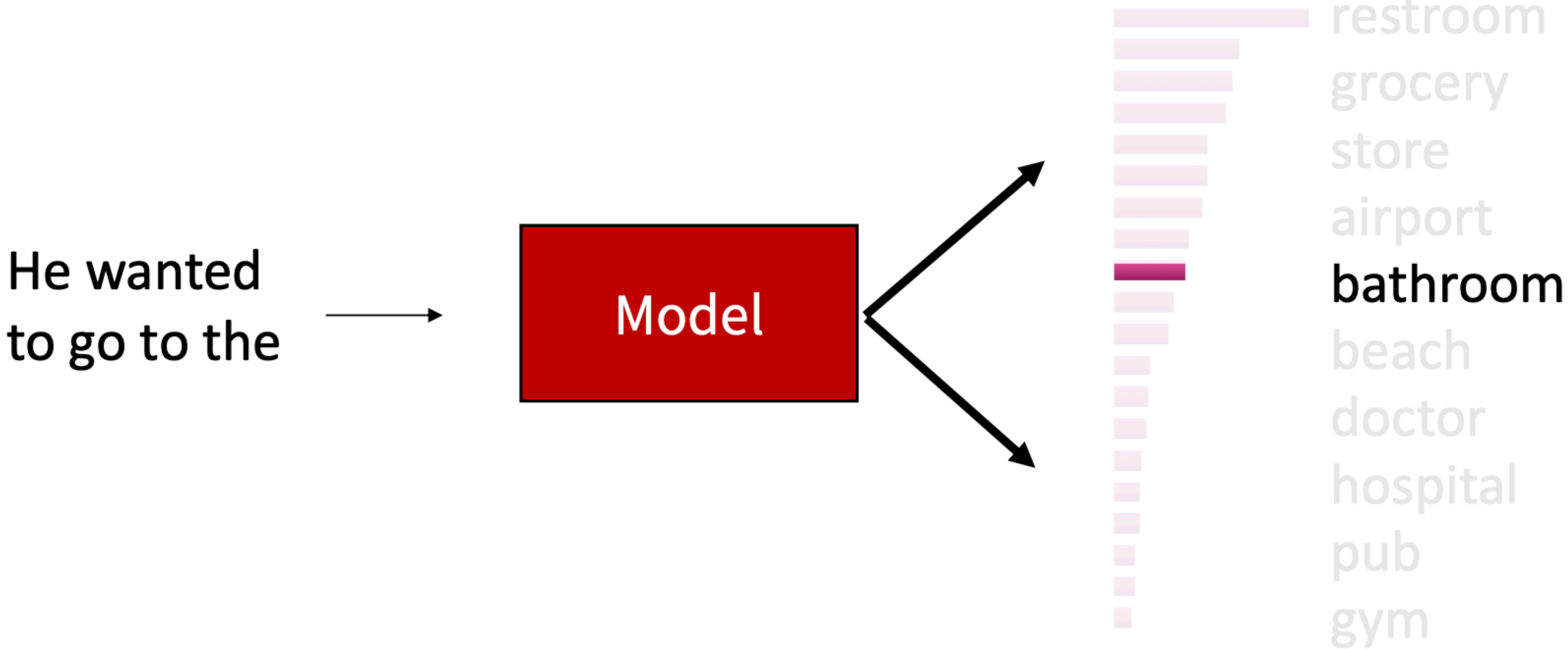
- Sample directly from P_t
- Still has access to the entire vocabulary
- But if the model distributions are of low quality, generations will be of low quality as well
- Often results in ill-formed generations



Pure / Ancestral Sampling

$$y_t \sim P_t(w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

- Sample directly from P_t
- Still has access to the entire vocabulary
- But if the model distributions are of low quality, generations will be of low quality as well
- Often results in ill-formed generations
  - No guarantee of fluency

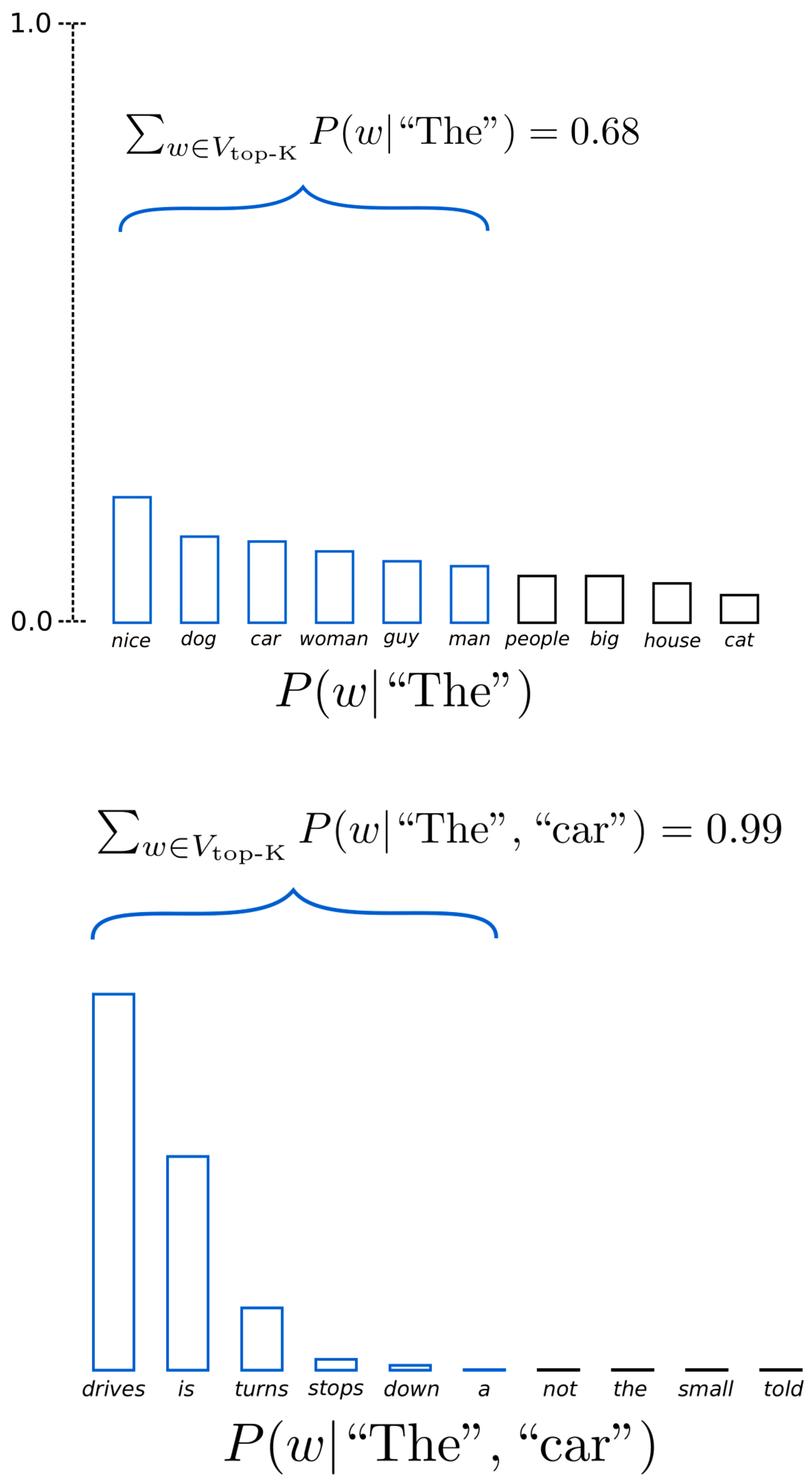


Fan et al., ACL 2018; Holtzman et al., ACL 2018

- Problem: Ancestral sampling makes every token in the vocabulary an option

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

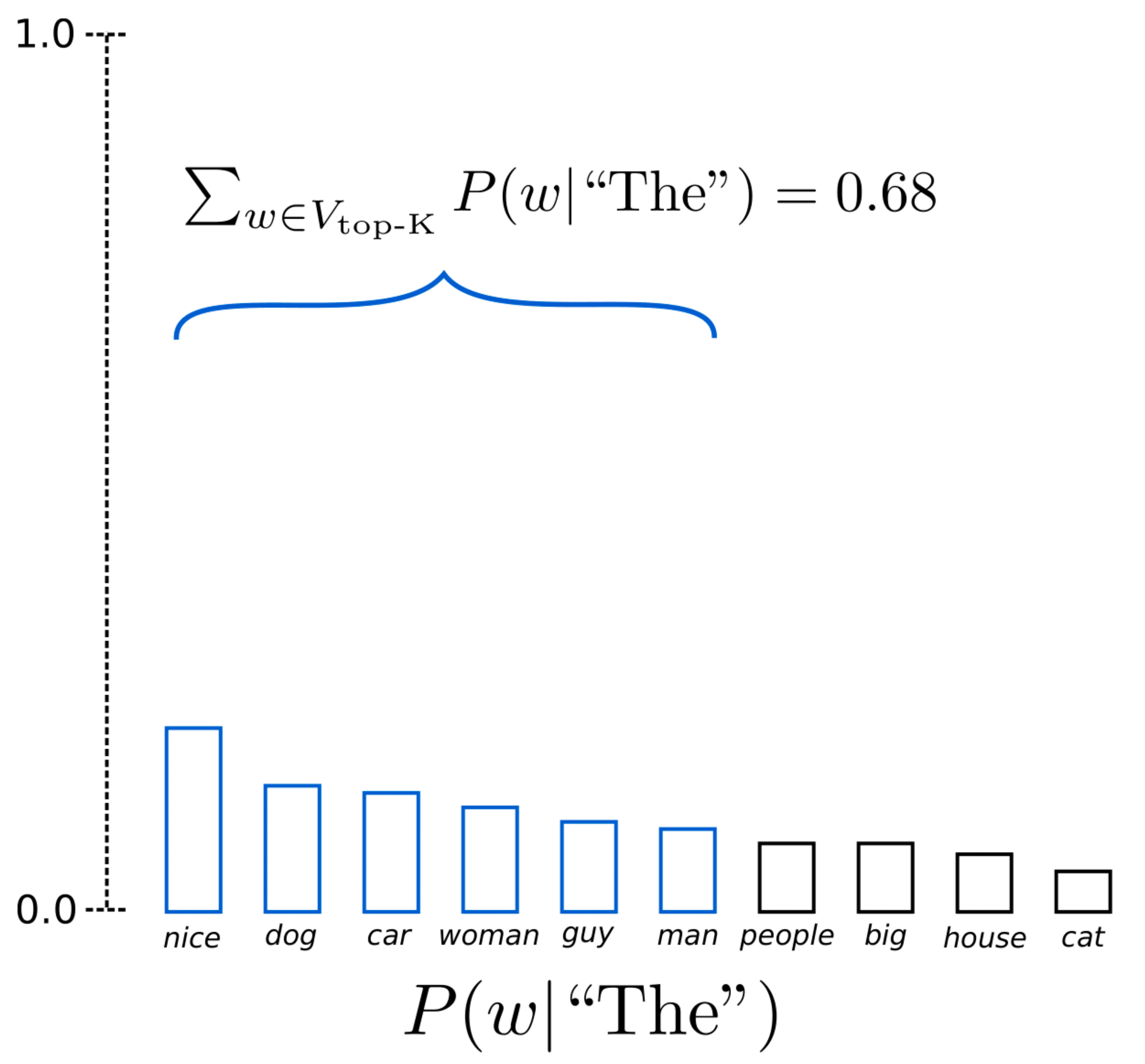
- Problem: Ancestral sampling makes every token in the vocabulary an option
  - Even if most of the probability mass in the distribution is over a limited set of options, the tail of the distribution could be very long and in aggregate have considerable mass

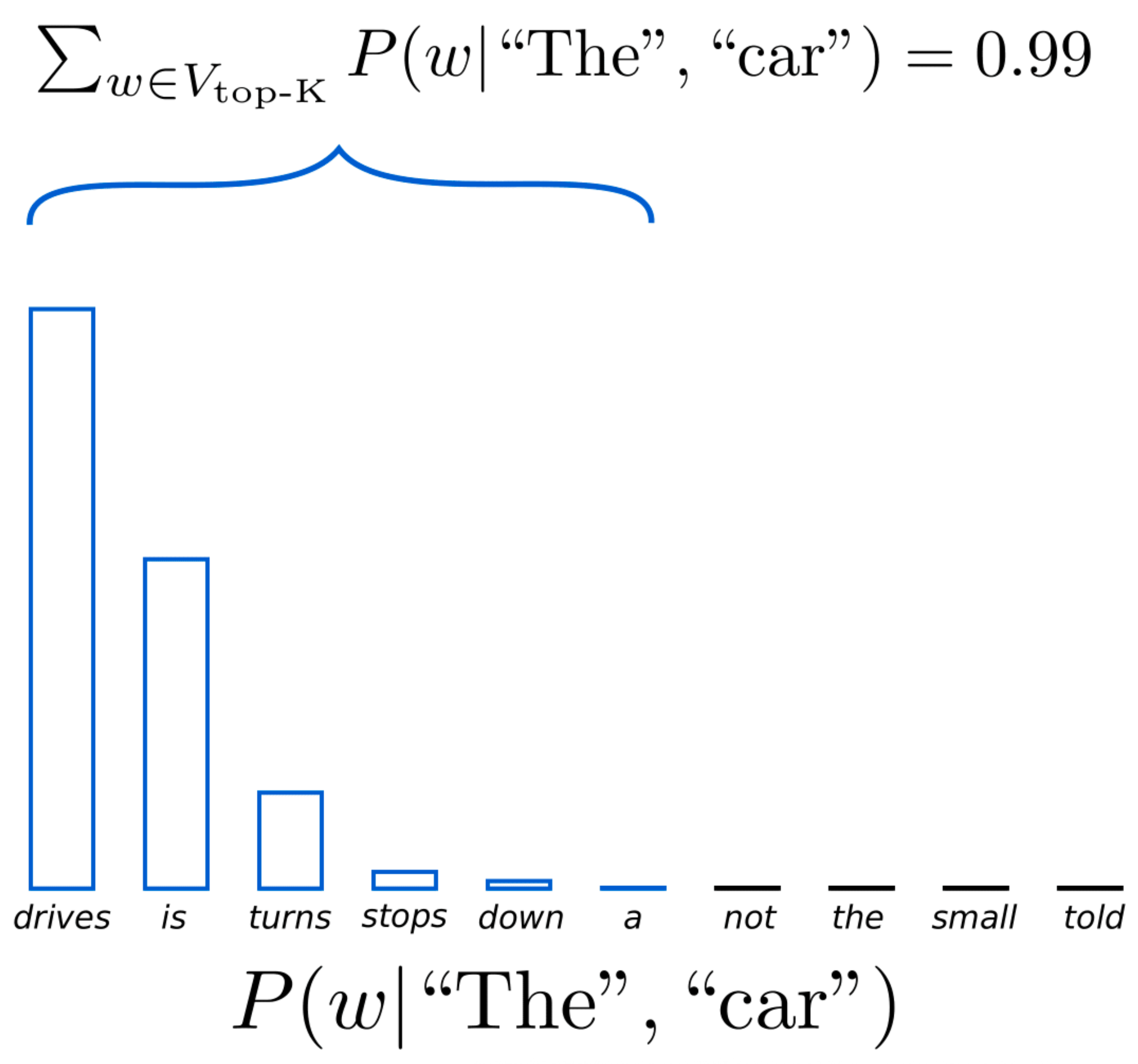


Fan et al., ACL 2018; Holtzman et al., ACL 2018

Image Source: Huggingface

- Problem: Ancestral 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 and in aggregate have considerable mass
  - Many tokens are probably really wrong in the current context. Yet, we give them individually a tiny chance to be selected.

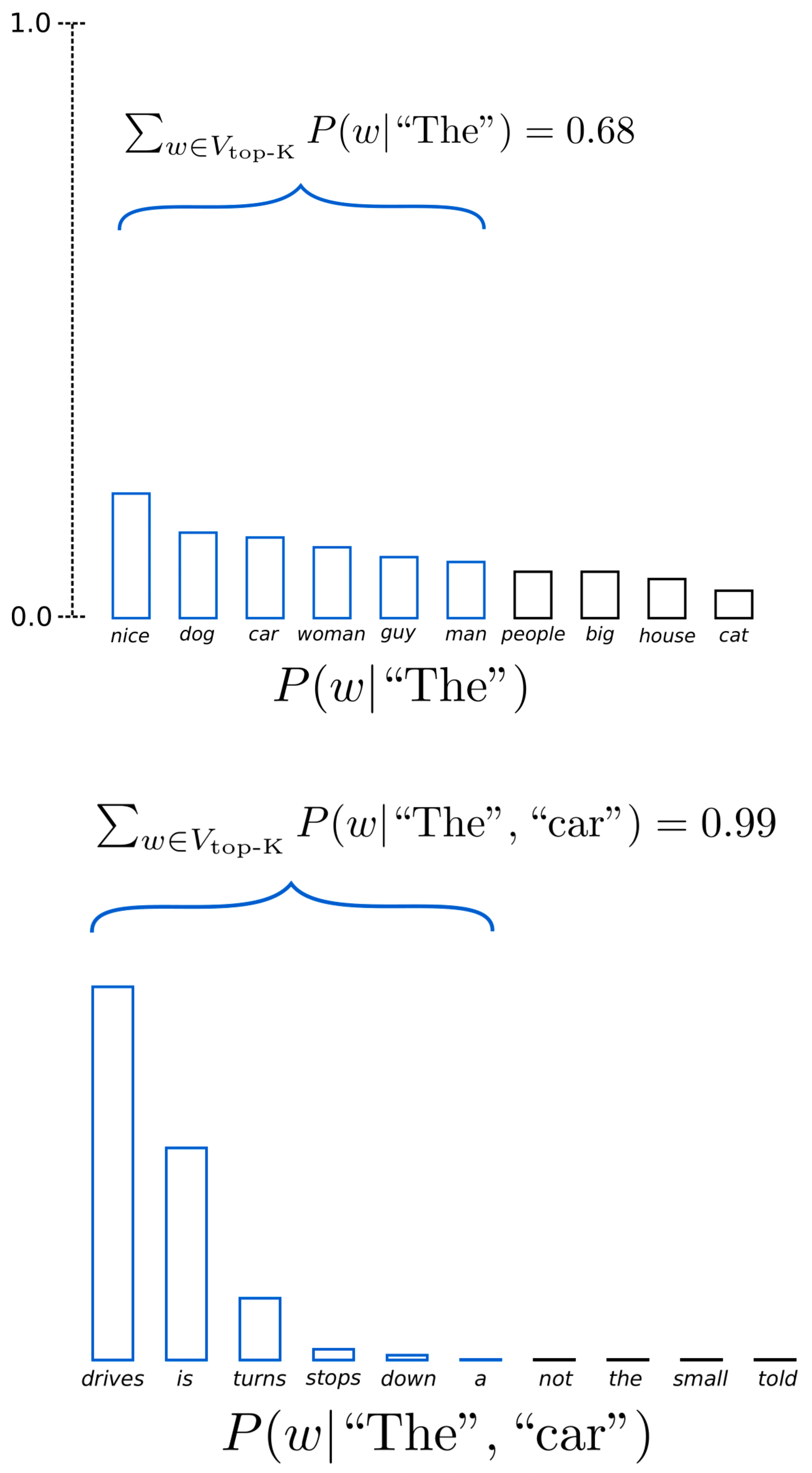




Fan et al., ACL 2018; Holtzman et al., ACL 2018

Image Source: Huggingface

- Problem: Ancestral 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 and in aggregate have considerable mass
  - Many tokens are probably really wrong in the current context. Yet, we give them individually a tiny chance to be selected.
  - But because there are many of them, we still give them as a group a high chance to be selected.

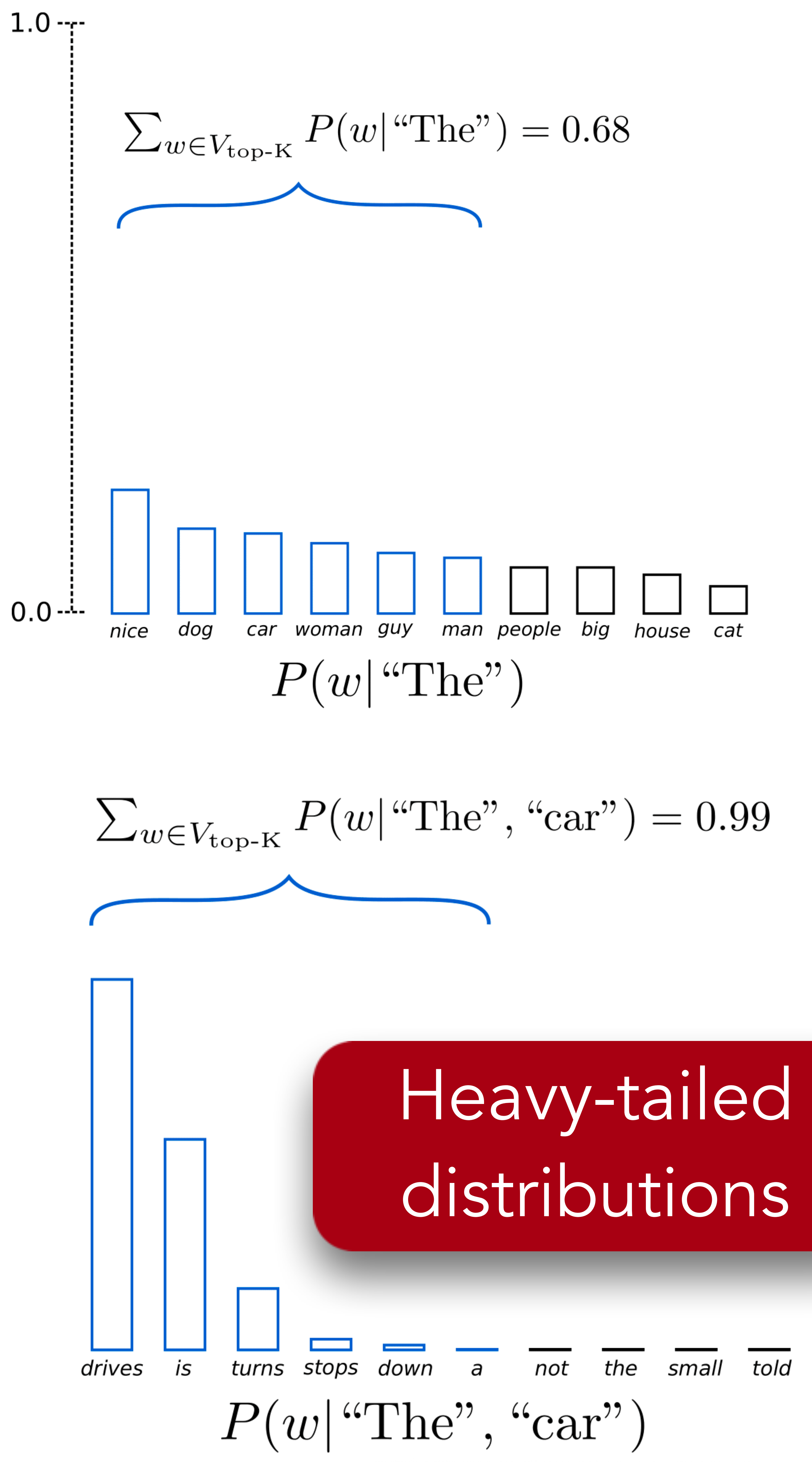


Fan et al., ACL 2018; Holtzman et al., ACL 2018

Image Source: Huggingface

USC Viterbi

- Problem: Ancestral 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 and in aggregate have considerable mass
  - Many tokens are probably really wrong in the current context. Yet, we give them individually a tiny chance to be selected.
  - But because there are many of them, we still give them as a group a high chance to be selected.

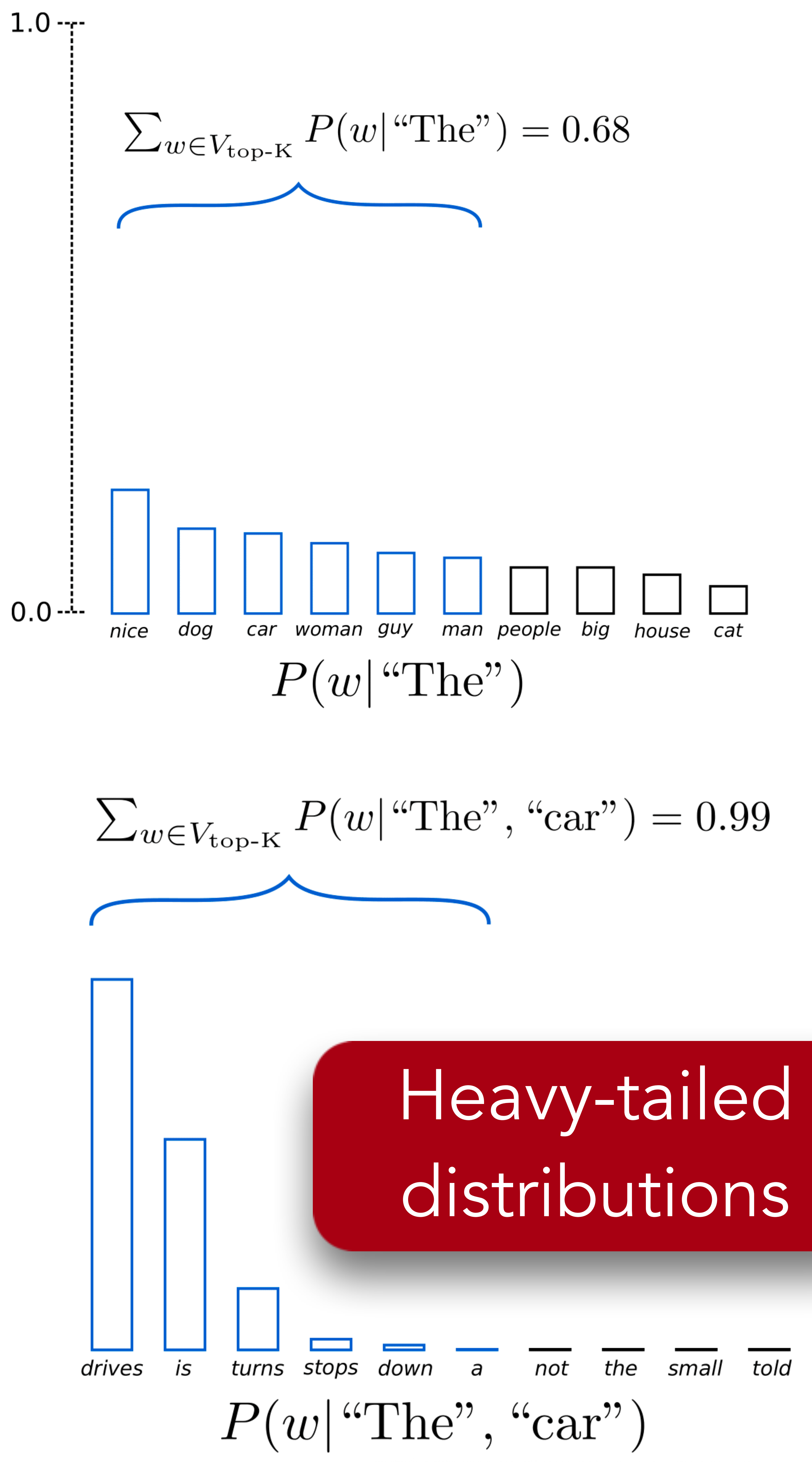


Heavy-tailed distributions

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

Image Source: Huggingface

- Problem: Ancestral 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 and in aggregate have considerable mass
  - Many tokens are probably really wrong in the current context. Yet, we give them individually a tiny chance to be selected.
  - But because there are many of them, we still give them as a group a high chance to be selected.
- Solution: Top-K sampling



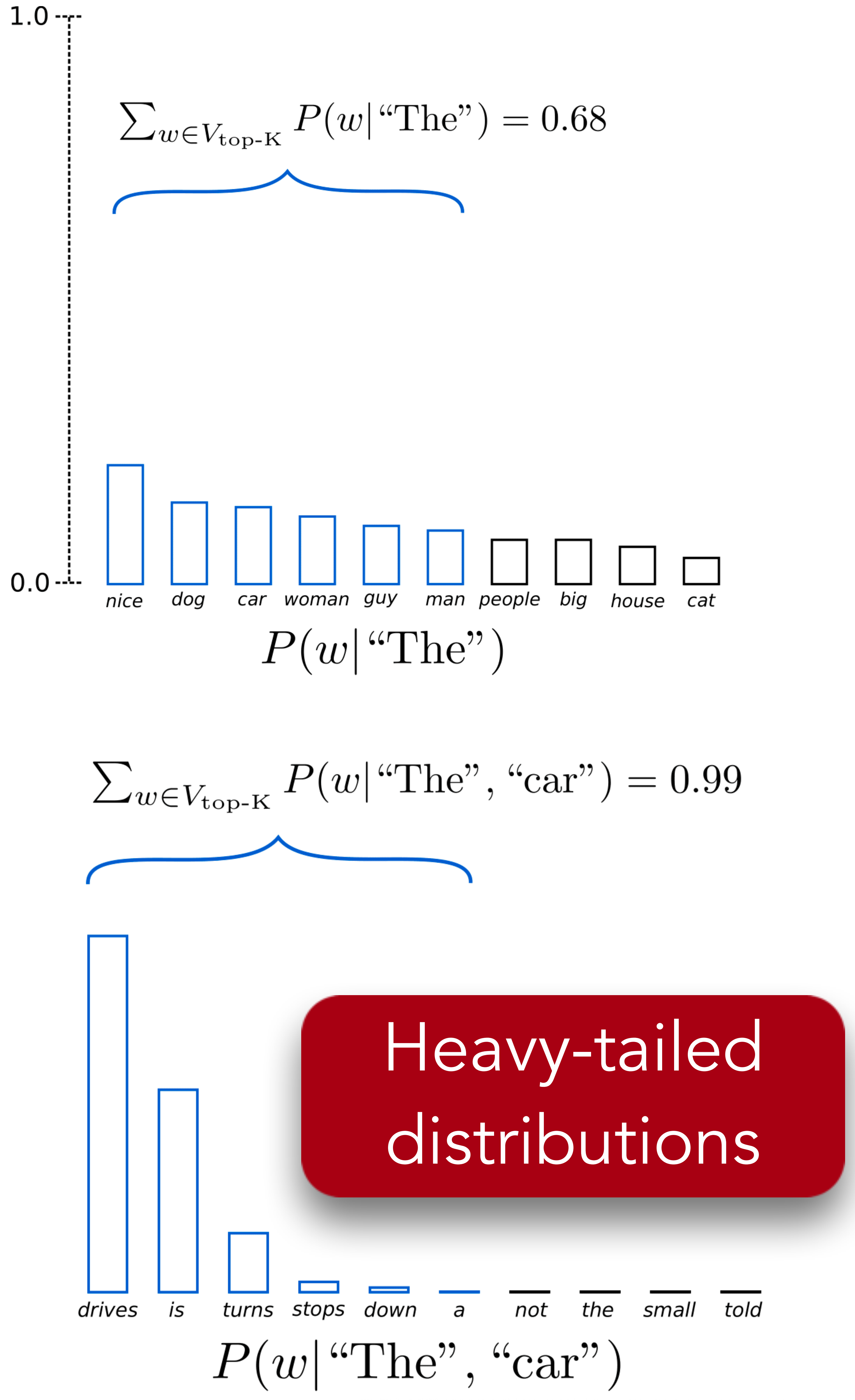
Heavy-tailed distributions

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

Image Source: Huggingface

Top-K Sampling

- Problem: Ancestral 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 and in aggregate have considerable mass
  - Many tokens are probably really wrong in the current context. Yet, we give them individually a tiny chance to be selected.
  - But because there are many of them, we still give 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

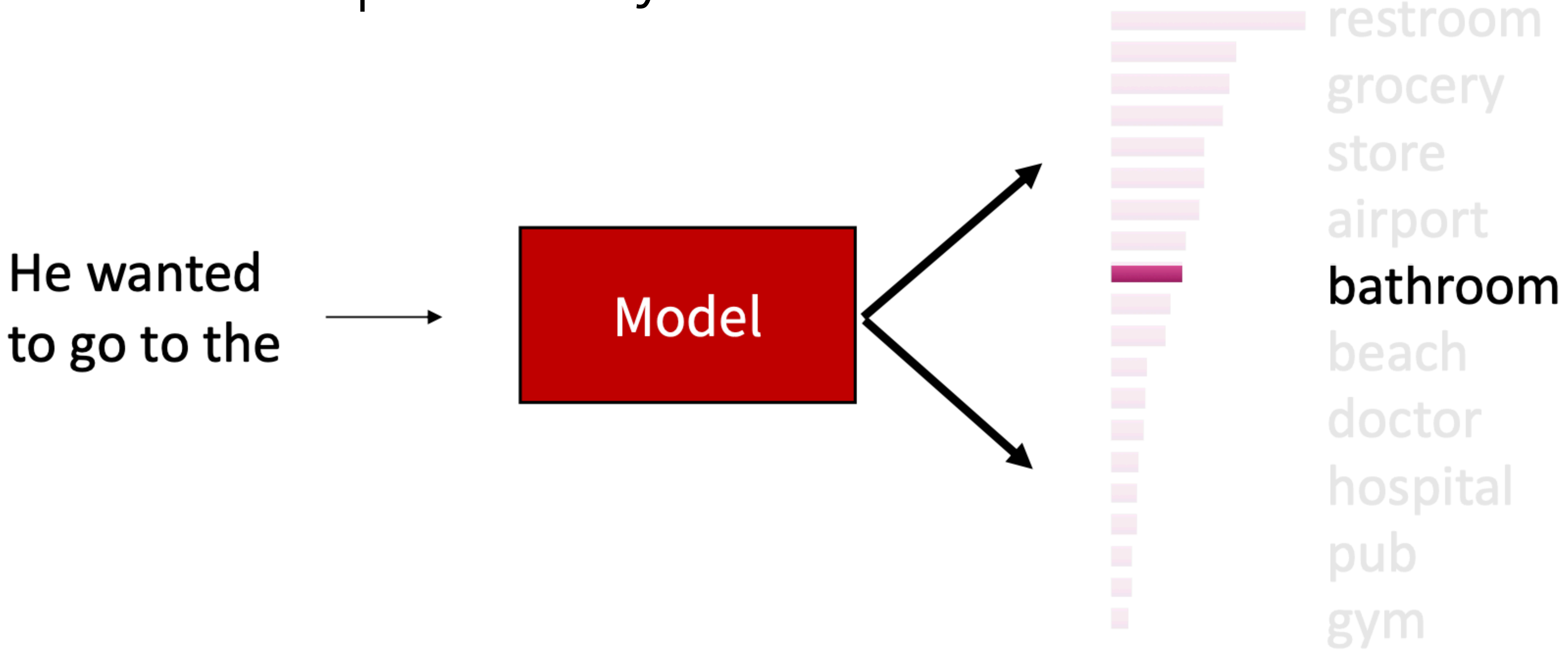
Image Source: Huggingface

Top-K Sampling: Value of K

- Solution: Top-K sampling
  - Only sample from the top K tokens in the probability distribution
  - Common values are $K = 50$

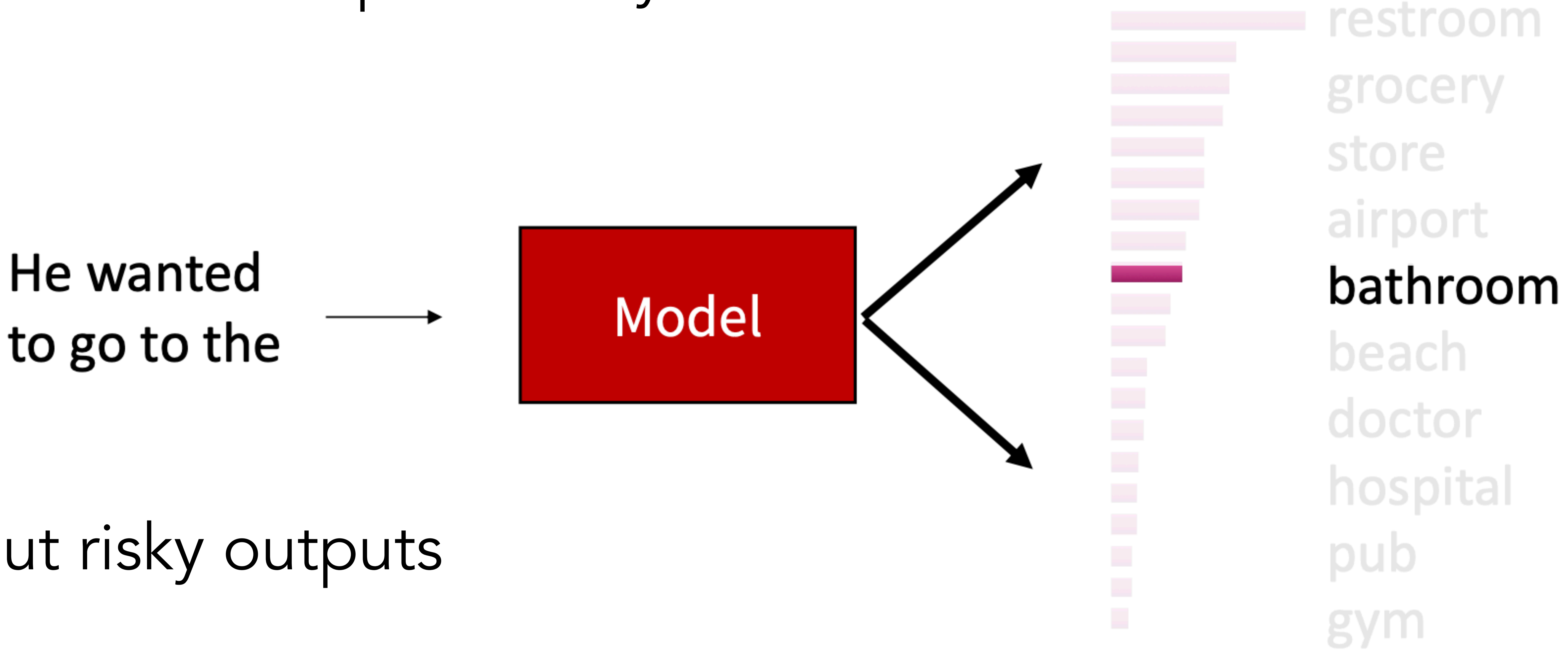
Top-K Sampling: Value of K

- Solution: Top-K sampling
  - Only sample from the top K tokens in the probability distribution
  - Common values are K = 50



Top-K Sampling: Value of K

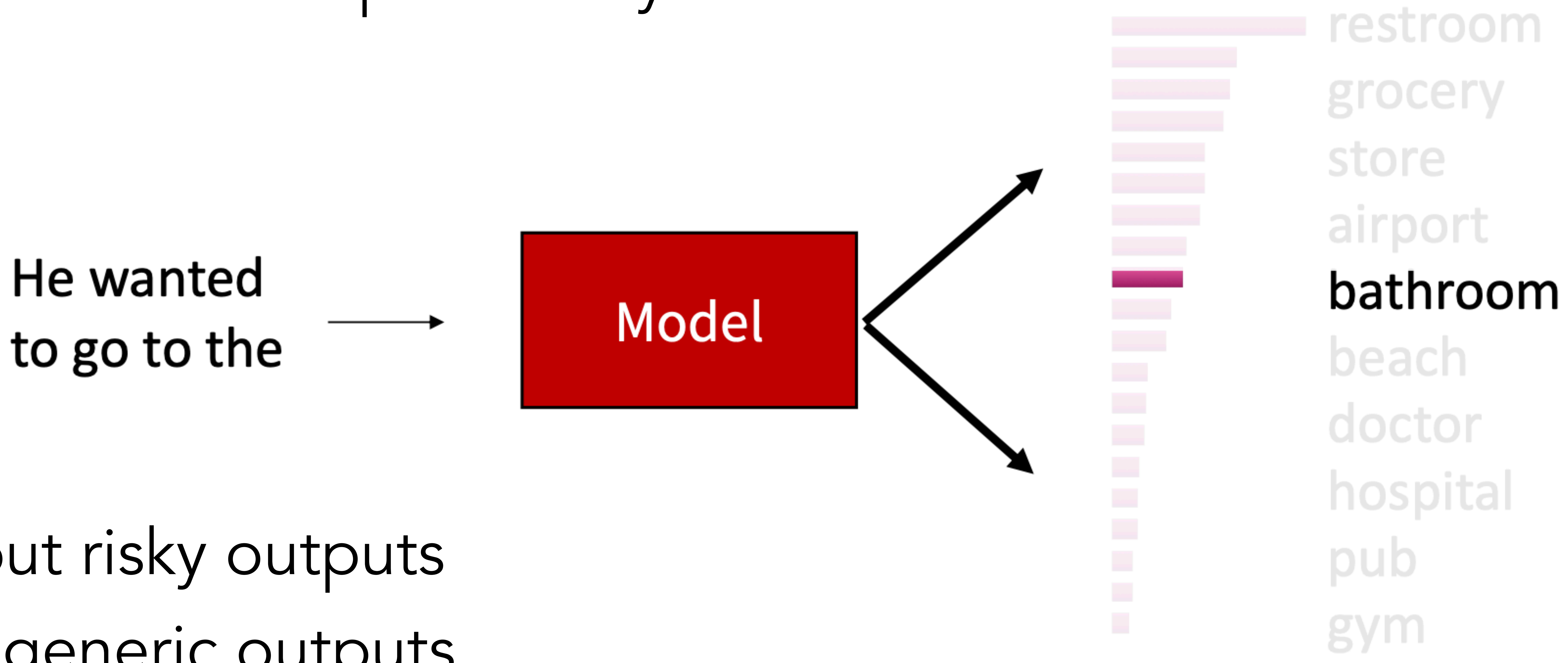
- Solution: Top-K sampling
  - Only sample from the top K tokens in the probability distribution
  - Common values are $K = 50$



- Increase K yields more diverse, but risky outputs

Top-K Sampling: Value of K

- Solution: Top-K sampling
  - Only sample from the top K tokens in the probability distribution
  - Common values are K = 50



- Increase K yields more diverse, but risky outputs
- Decrease K yields more safe but generic outputs

Top-K Sampling: Issues

Top-K sampling can cut off too quickly

Image Source: Holtzmann et al., 2019

Top-K Sampling: Issues

Top-K sampling can cut off too quickly

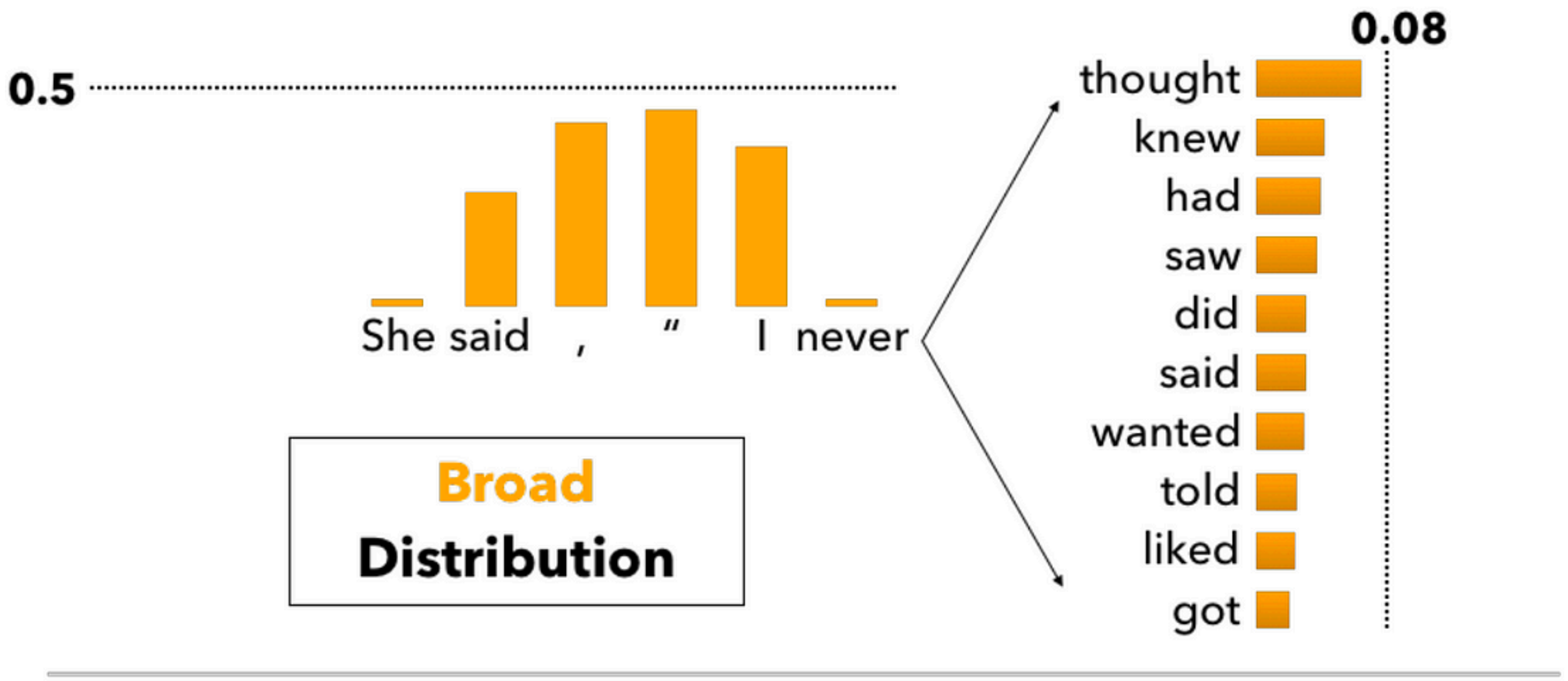


Image Source: Holtzmann et al., 2019

USC Viterbi

Top-K Sampling: Issues

Top-K sampling can cut off too quickly

Top-K sampling can also cut off too slowly!

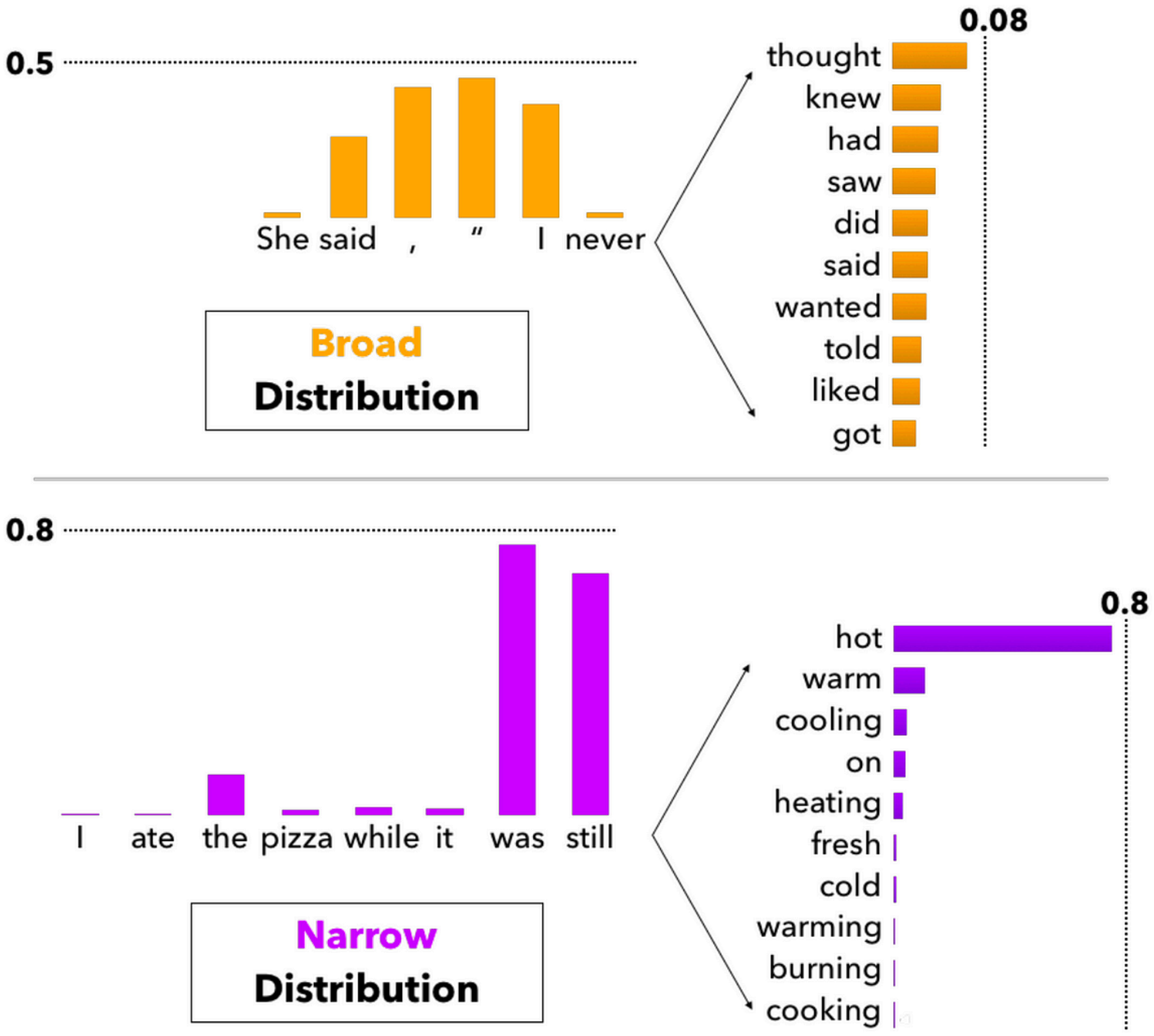


Image Source: Holtzmann et al., 2019

Top-K Sampling: Issues

Top-K sampling can cut off too quickly

Top-K sampling can also cut off too slowly!

We can do better than having one-size-fits-all: a fixed K for all contexts

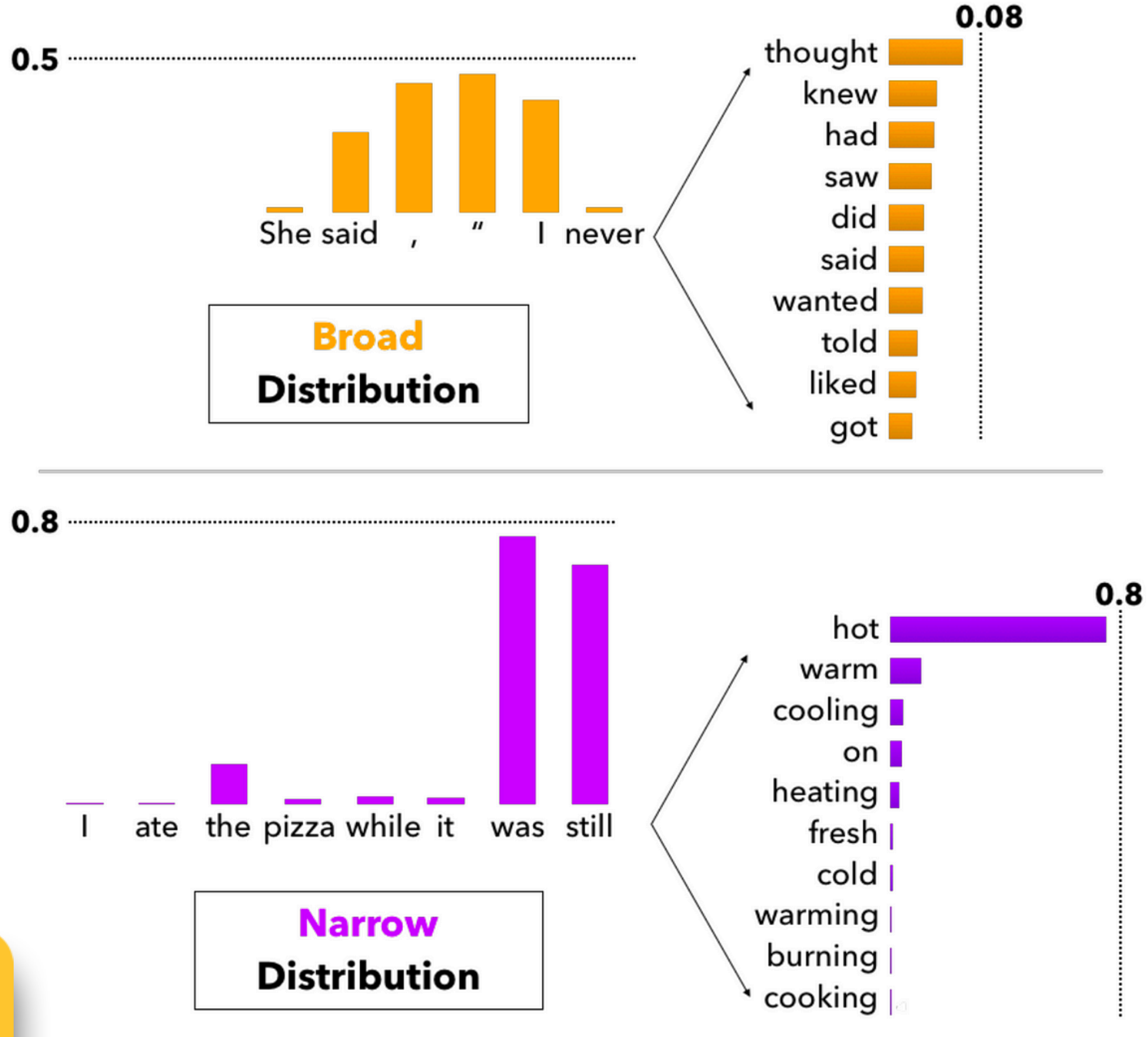


Image Source: Holtzmann et al., 2019

Holtzman et al., ICLR 2020

- Problem: The probability distributions we sample from are dynamic

Holtzman et al., ICLR 2020

- Problem: The probability distributions we sample from are dynamic
  - When the distribution P_t is flatter, a limited K removes many viable options

Holtzman et al., ICLR 2020

- 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

Modern Decoding: 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: Nucleus Sampling / Top-P sampling

Holtzman et al., ICLR 2020

Modern Decoding: 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: Nucleus Sampling / Top-P sampling
  - Sample from all tokens in the top P cumulative probability mass (i.e., where mass is concentrated)

Holtzman et al., ICLR 2020

Modern Decoding: 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

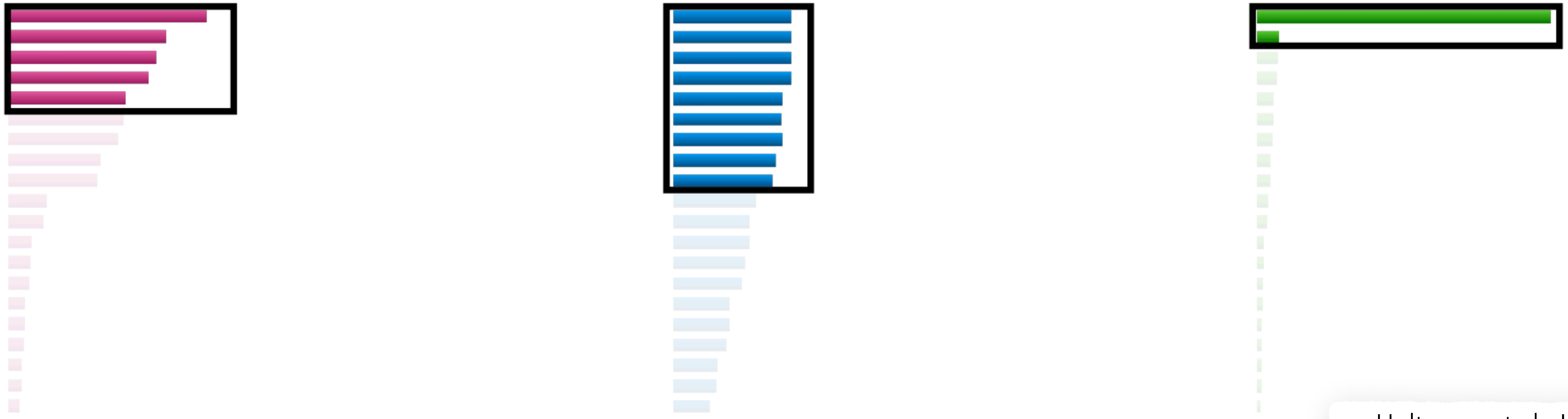
- Solution: Nucleus Sampling / Top-P sampling
  - Sample from all tokens in the top P cumulative probability mass (i.e., where mass is concentrated)
  - Varies K depending on the uniformity of P_t

Holtzman et al., ICLR 2020

Nucleus (Top-P) Sampling

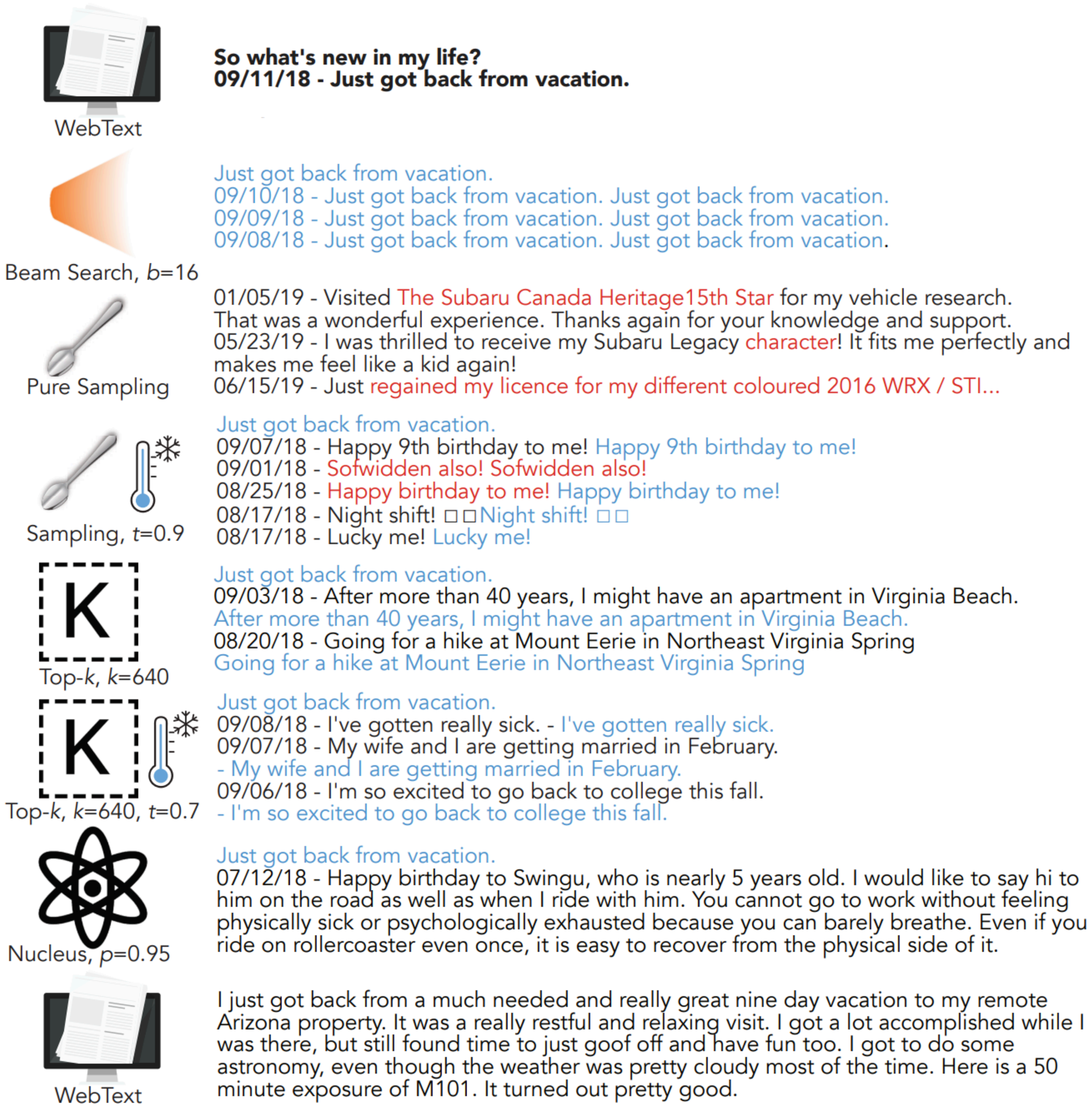
- Solution: Top-P sampling
  - Sample from all tokens in the top P cumulative probability mass (i.e., where mass is concentrated)
  - Varies K depending on the uniformity of P_t

$P_t^1(y_t = w \mid \{y\}_{<t})$ $\quad$ $P_t^2(y_t = w \mid \{y\}_{<t})$ $\quad$ $P_t^3(y_t = w \mid \{y\}_{<t})$



Holtzman et al., ICLR 2020

Comparing different decoding algorithms

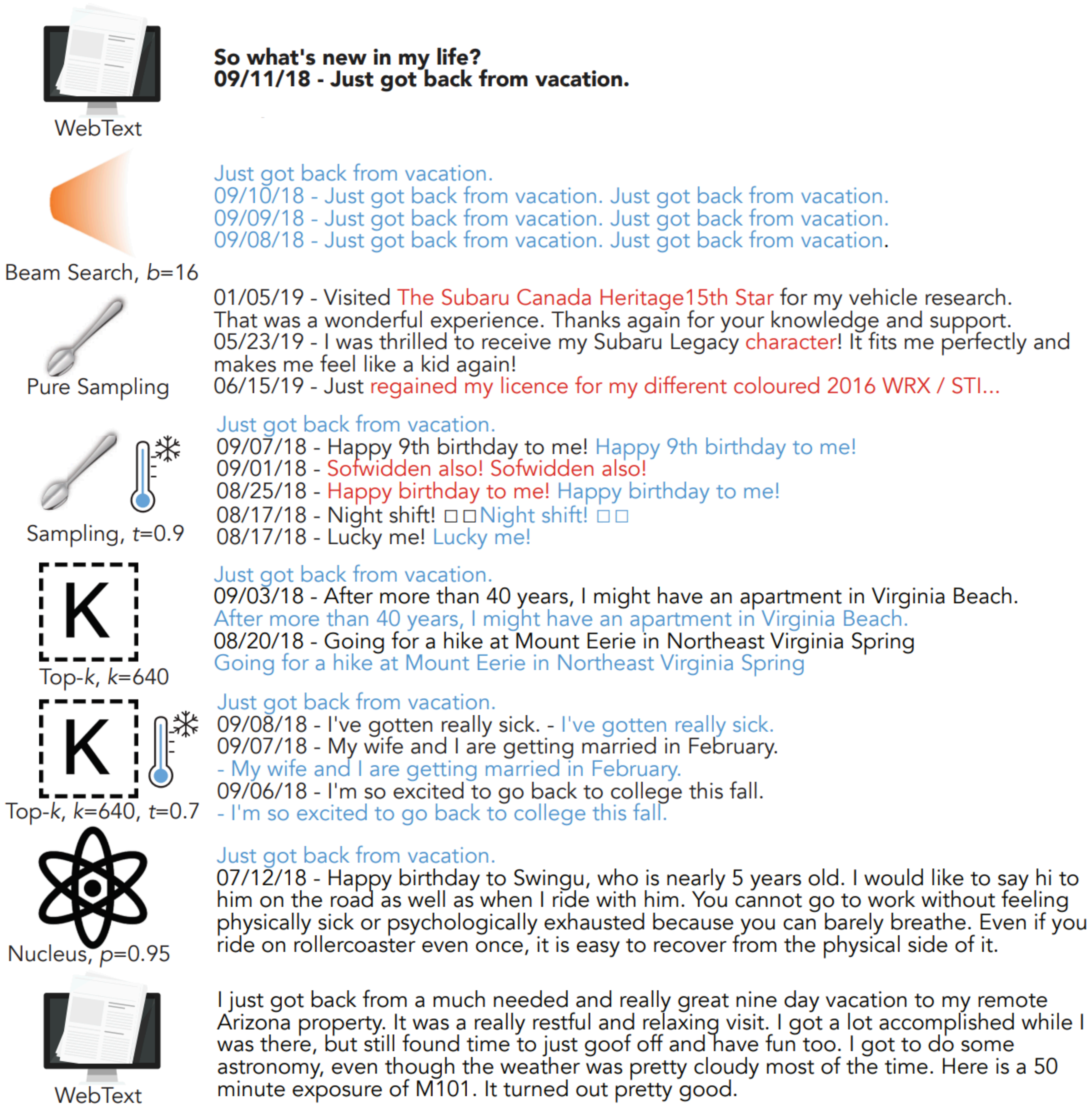


Holtzman et al., ICLR 2020

USC Viterbi

Comparing different decoding algorithms

- Generate text to continue a given context
  - Open-ended generation

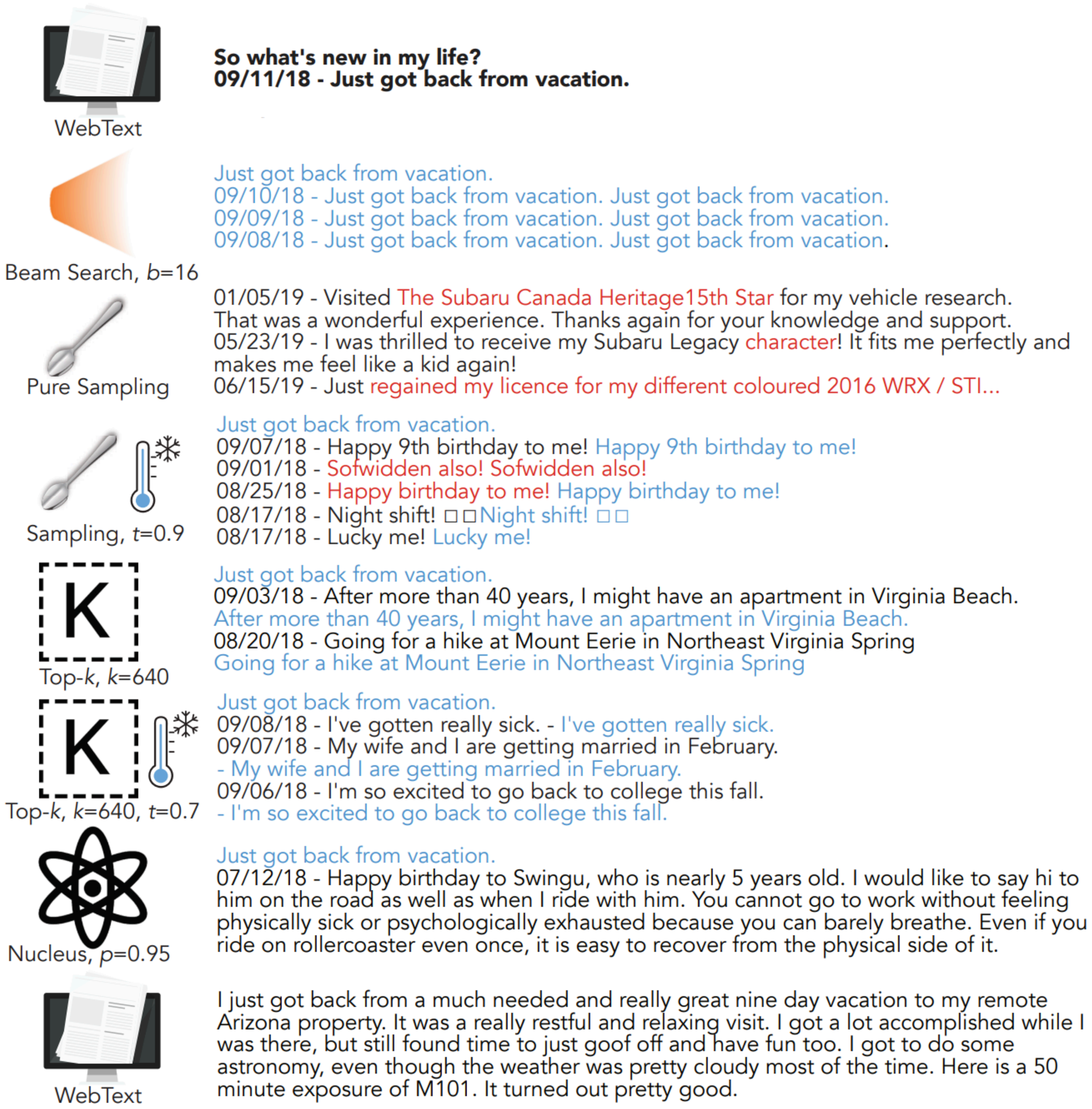


Holtzman et al., ICLR 2020

USC Viterbi

Comparing different decoding algorithms

- Generate text to continue a given context
  - Open-ended generation
- Same decoding algorithms are also useful for close-ended generation tasks



Holtzman et al., ICLR 2020

Temperature Scaling

$$P(y_t = w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

Temperature Scaling

$$P(y_t = w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

- Recall: On timestep t, the model computes a prob distribution P_t by applying the softmax function to a vector of scores $s \in \mathbb{R}^{|V|}$

USC Viterbi

Temperature Scaling

$$P(y_t = w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

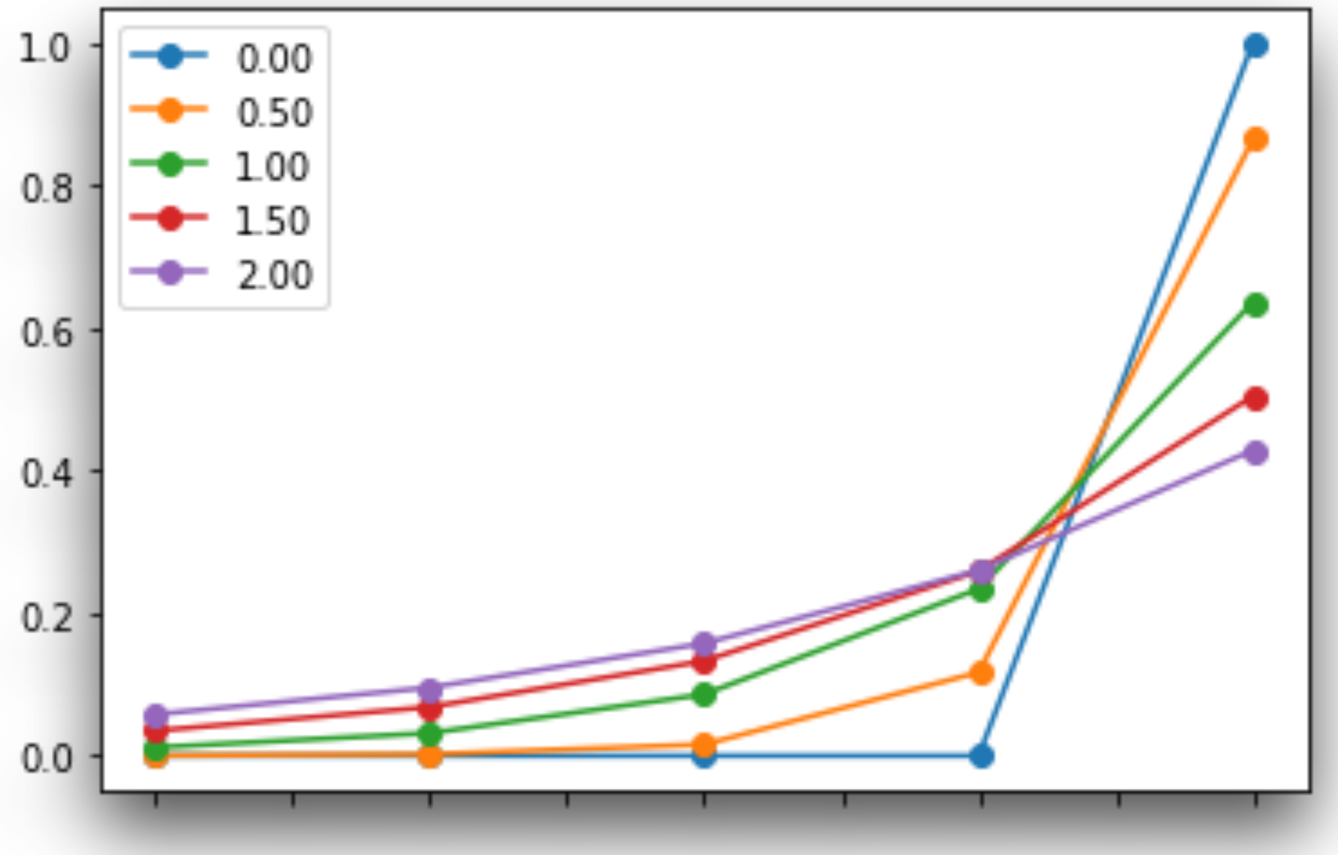
- Recall: On timestep t, the model computes a prob distribution P_t by applying the softmax function to a vector of scores $s \in \mathbb{R}^{|V|}$
- We can apply a temperature hyperparameter τ to the softmax to rebalance P_t

$$P(y_t = w) = \frac{\exp(S_w/\tau)}{\sum_{v \in V} \exp(S_v/\tau)}$$

Temperature Scaling

$$P(y_t = w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

- Recall: On timestep t, the model computes a prob distribution P_t by applying the softmax function to a vector of scores $s \in \mathbb{R}^{|V|}$
- We can apply a temperature hyperparameter τ to the softmax to rebalance P_t



$$P(y_t = w) = \frac{\exp(S_w/\tau)}{\sum_{v \in V} \exp(S_v/\tau)}$$

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

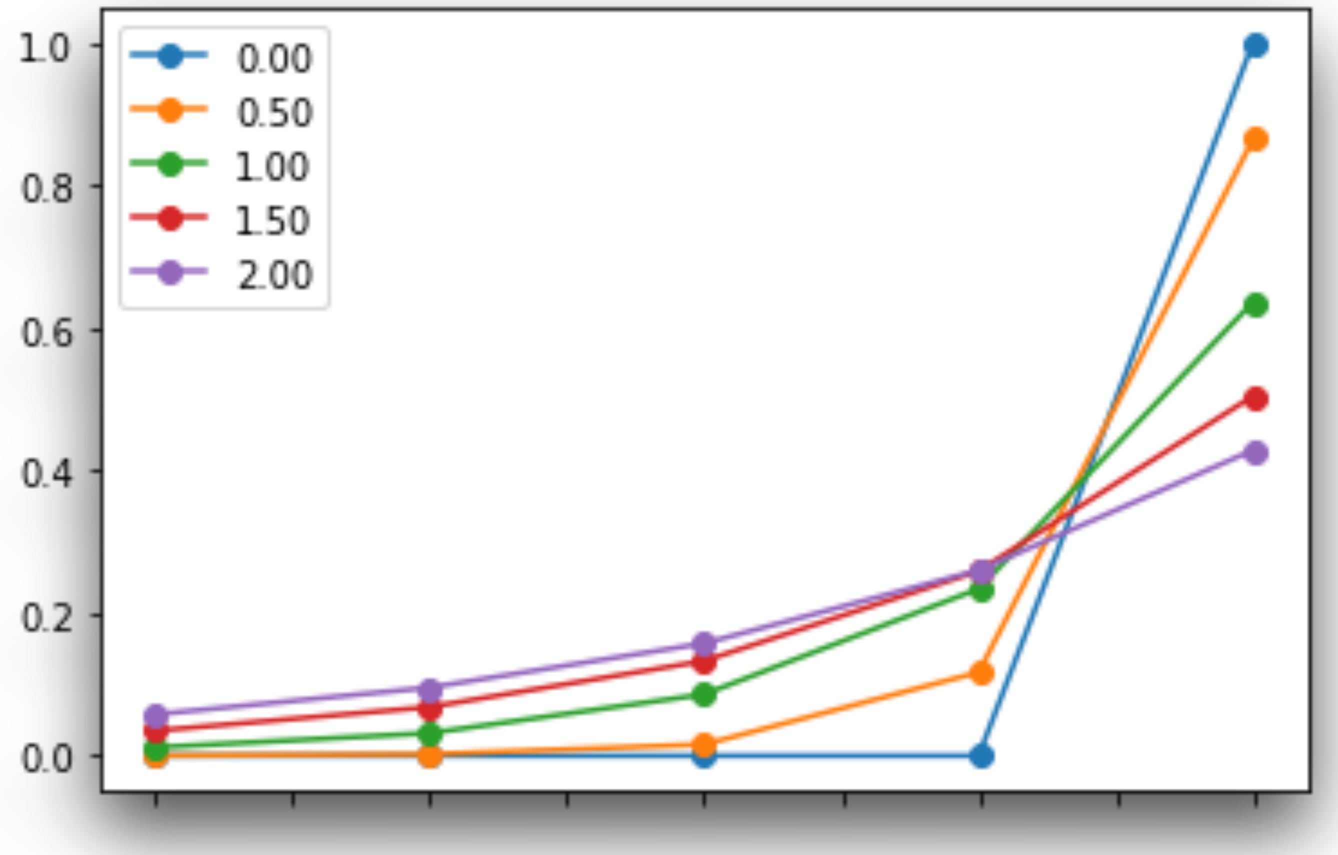
USC Viterbi

Temperature Scaling

$$P(y_t = w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

- Recall: On timestep t, the model computes a prob distribution P_t by applying the softmax function to a vector of scores $s \in \mathbb{R}^{|V|}$
- We can apply a temperature hyperparameter τ to the softmax to rebalance P_t

$$P(y_t = w) = \frac{\exp(S_w/\tau)}{\sum_{v \in V} \exp(S_v/\tau)}$$



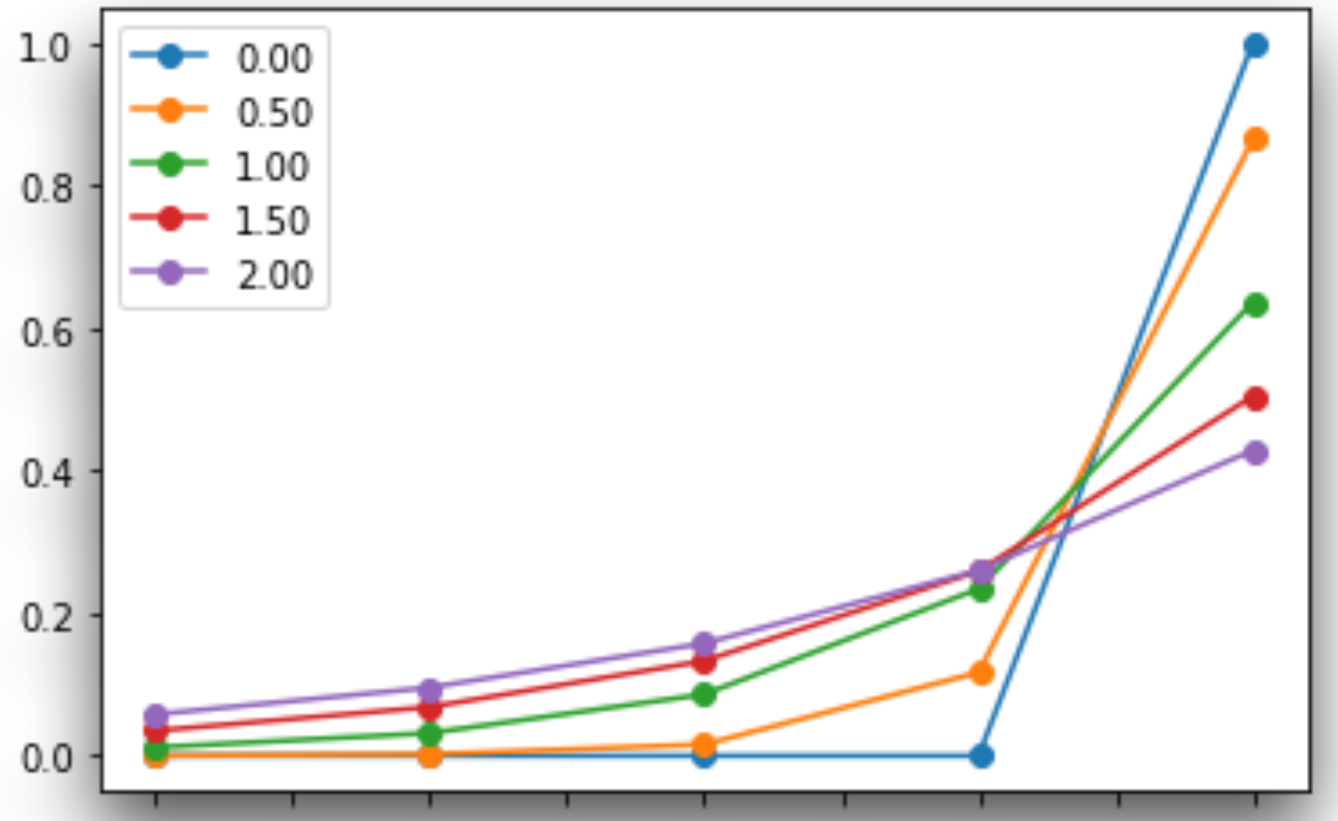
- Raise the temperature $\tau > 1$: P_t becomes more uniform
  - More diverse output (probability is spread around vocab)
- Lower the temperature $\tau < 1$: P_t becomes more spiky
  - Less diverse output (probability is concentrated on top words)

Temperature Scaling

$$P(y_t = w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

- Recall: On timestep t, the model computes a prob distribution P_t by applying the softmax function to a vector of scores $s \in \mathbb{R}^{|V|}$
- We can apply a temperature hyperparameter τ to the softmax to rebalance P_t

$$P(y_t = w) = \frac{\exp(S_w / \tau)}{\sum_{v \in V} \exp(S_v / \tau)}$$



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

Temperature is a hyperparameter for decoding: It can be tuned for both beam search and sampling.

Modern Decoding: Takeaways

Modern Decoding: Takeaways

- Natural language distributions are very peaky but the softmax function assigns probabilities to all tokens in the vocabulary

Modern Decoding: Takeaways

- Natural language distributions are very peaky but the softmax function assigns probabilities to all tokens in the vocabulary
- Hence we need approaches to truncate / modify the softmax distribution

Modern Decoding: Takeaways

- Natural language distributions are very peaky but the softmax function assigns probabilities to all tokens in the vocabulary
- Hence we need approaches to truncate / modify the softmax distribution
  - Ancestral, Top-k, Top-p (Nucleus), Temperature

USC Viterbi

Modern Decoding: Takeaways

- Natural language distributions are very peaky but the softmax function assigns probabilities to all tokens in the vocabulary
- Hence we need approaches to truncate / modify the softmax distribution
  - Ancestral, Top-k, Top-p (Nucleus), Temperature
- Some properties of the softmax function make truncation based decoding necessary

CLOSING THE CURIOUS CASE OF NEURAL TEXT DEGENERATION

Matthew Finlayson
University of Southern California
mfinlays@usc.edu

John Hewitt
Stanford University
johnhew@cs.stanford.edu

Alexander Koller
Saarland University
koller@coli.uni-saarland.de

Swabha Swayamdipta
University of Southern California
swabhas@usc.edu

Ashish Sabharwal
The Allen Institute for AI
ashishs@allenai.org

Modern Decoding: Takeaways

- Natural language distributions are very peaky but the softmax function assigns probabilities to all tokens in the vocabulary
- Hence we need approaches to truncate / modify the softmax distribution
  - Ancestral, Top-k, Top-p (Nucleus), Temperature
- Some properties of the softmax function make truncation based decoding necessary

CLOSING THE CURIOUS CASE OF NEURAL TEXT DEGENERATION

Matthew Finlayson
University of Southern California
mfinlays@usc.edu

John Hewitt
Stanford University
johnhew@cs.stanford.edu

Alexander Koller
Saarland University
koller@coli.uni-saarland.de

Swabha Swayamdipta
University of Southern California
swabhas@usc.edu

Ashish Sabharwal
The Allen Institute for AI
ashishs@allenai.org

Next Class: Evaluating Generations (Me), Prompting and Instruction Tuning (Guest Lecture)

Quiz 5