



A Hot Take on Sampling from Probabilistic Text Generators

Ryan Cotterell @



ETH zürich



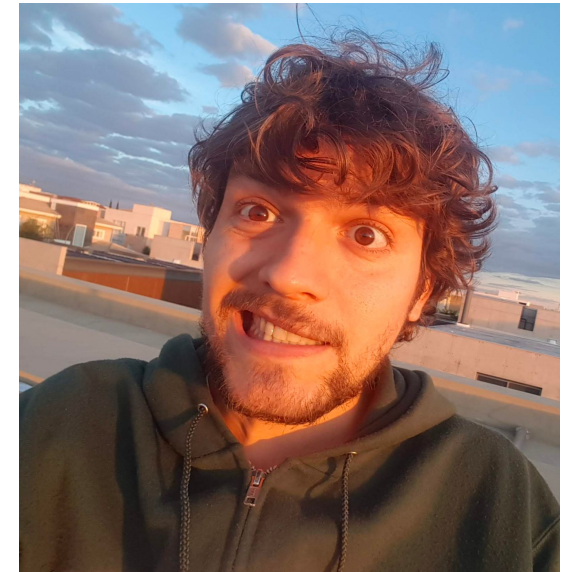
Thanks to my amazing collaborators!



Clara Meister
ETH Zürich
(2nd-year PhD student)



Gian Wiher
ETH Zürich
(MSc student)



Tiago Pimentel
Cambridge
(3rd-year PhD student)

Probabilistic Text Generation



Write With Transformer `gpt2` ⓘ

 Shuffle initial text  Trigger autocomplete or `tab`
Select suggestion `↑` `↓` and `enter` Cancel suggestion `esc`

Save & Publish 

The chicken crossed the road because



HUGGING FACE

the chicken had to cross a busy road,

it was so slow.

they didn't want to have to worry about

Probabilistic Text Generation



Write With Transformer `gpt2` ⓘ

↻ Shuffle initial text

⏵ Trigger autocomplete or `tab`

Select suggestion `↑` `↓` and `enter`

Cancel suggestion `esc`

Save & Publish 

The chicken crossed the road because



HUGGING FACE

the chicken had to cross a busy road,

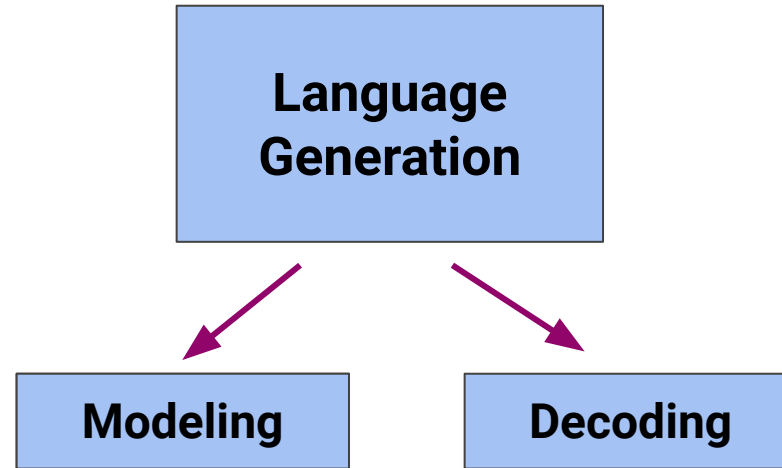
it was so slow.

they didn't want to have to worry about

What is a good algorithm to generate this text?

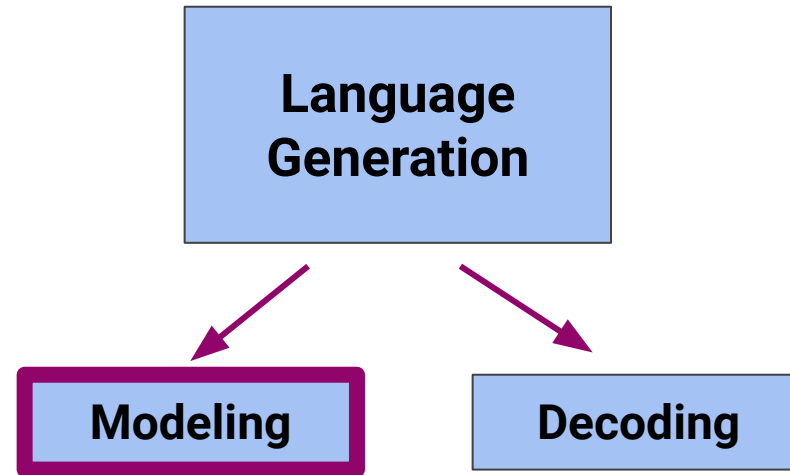


Probabilistic Text Generation



We can view probabilistic natural language generation as a two part problem

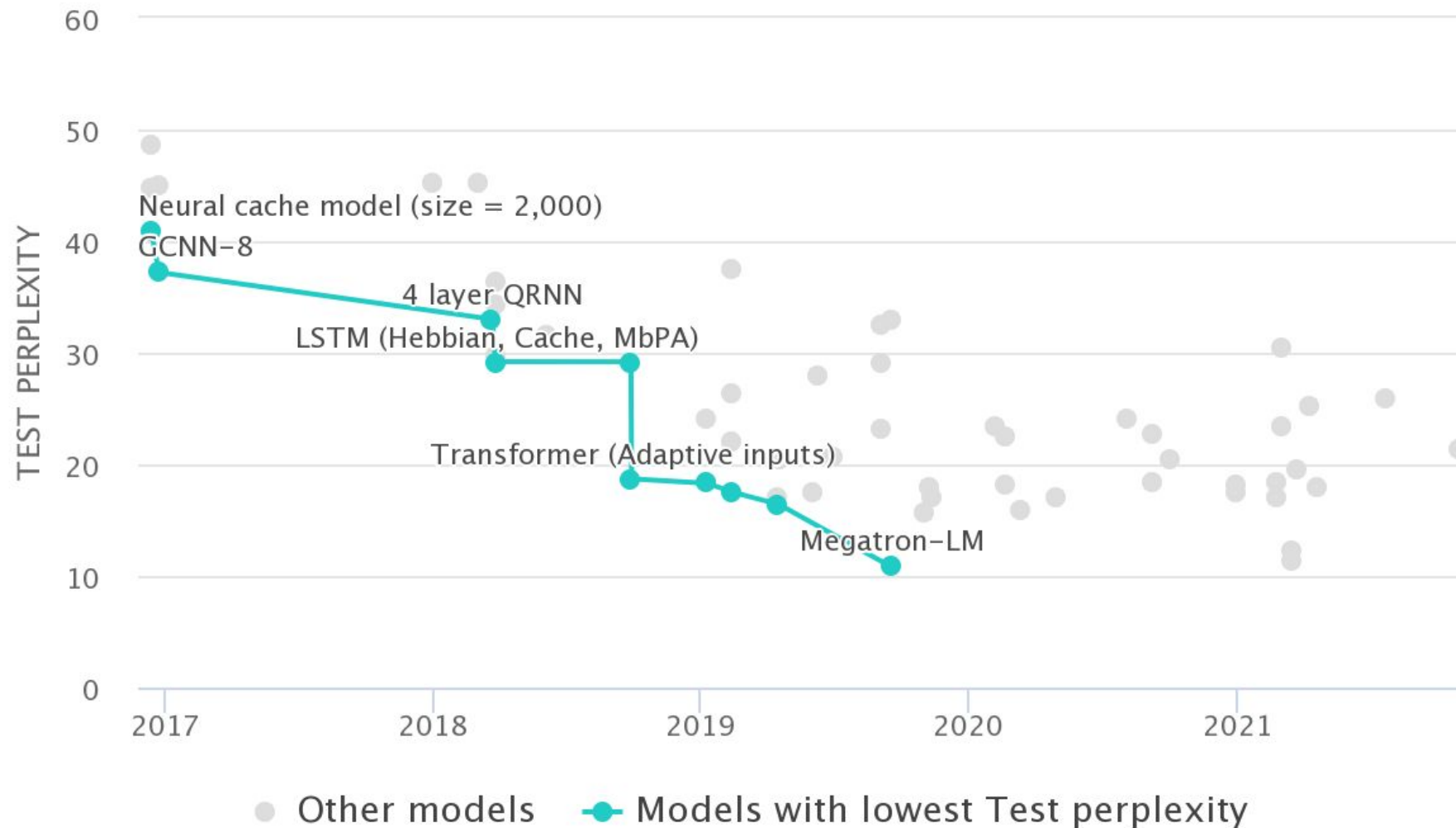
Probabilistic Text Generation: Modeling



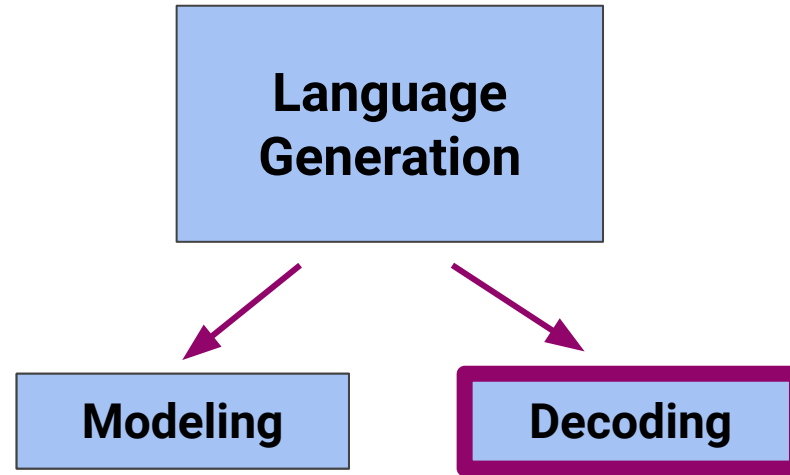
Modeling: Which probability distribution should we generate text from?

- Left-to-right “causal” language model?
- Cloze language model?
- Globally versus locally normalized models?

Probabilistic Text Generation: Modeling



Probabilistic Text Generation: Decoding



Decoding: Which *decoding strategy* should we use to generate the text?

- Ancestral sampling?
- Beam search?
- Dynamic programming (might be slow!)?

This Talk Focuses on Decoding!
(It's surprising how much the decoding strategy matters!)

The Mechanics of Decoding

Let's focus on a traditional left-to-right language model that decomposes as follows

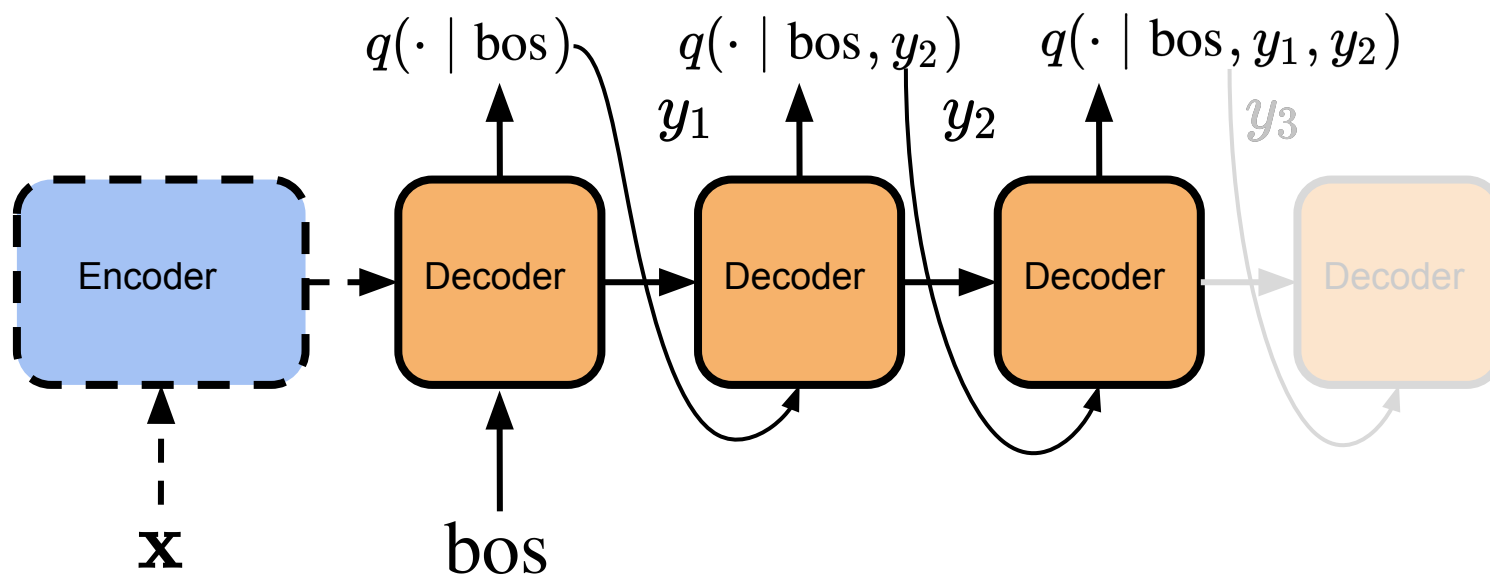
$$q(\mathbf{y}) = \prod_{t=1}^T q(y_t \mid y_1, \dots, y_{t-1})$$

The Mechanics of Decoding

Let's focus on a traditional left-to-right language model that decomposes as follows

$$q(\mathbf{y}) = \prod_{t=1}^T q(y_t \mid y_1, \dots, y_{t-1})$$

We then generate y_1 according to $q(\cdot \mid \text{bos})$, y_2 according to $q(\cdot \mid \text{bos}, y_1)$

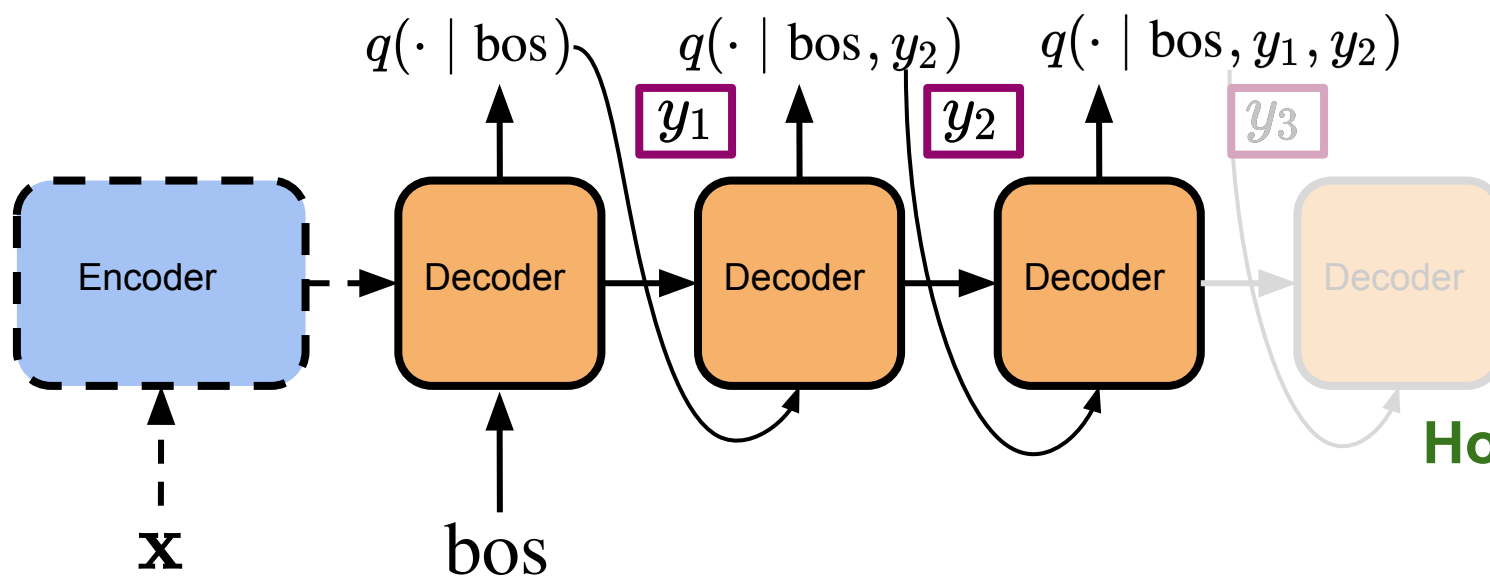


The Mechanics of Decoding

Let's focus on a traditional left-to-right language model that decomposes as follows

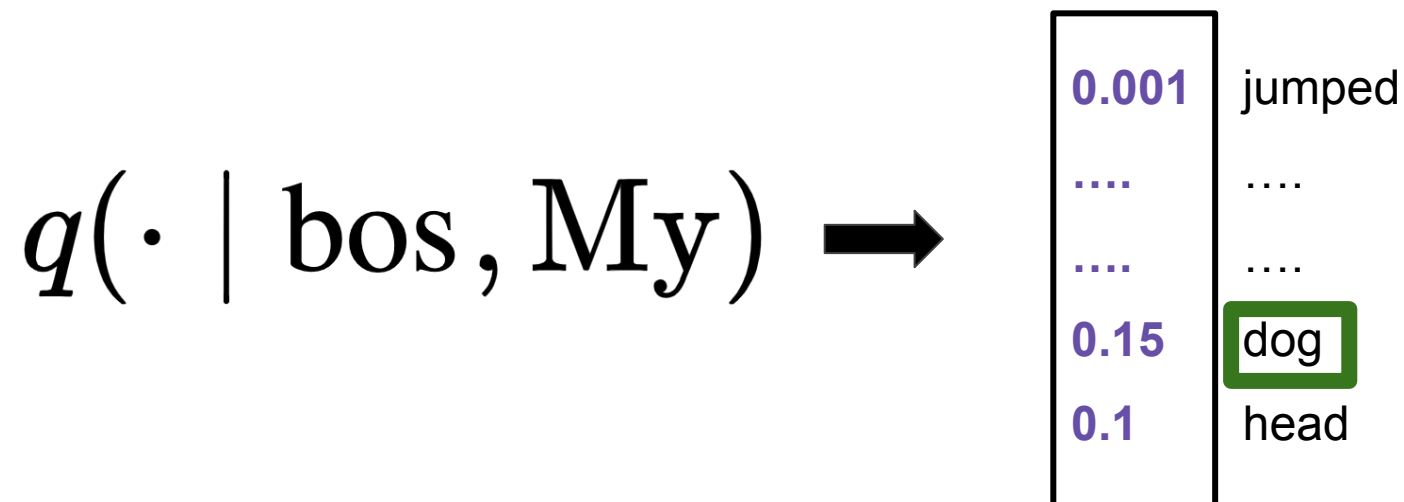
$$q(\mathbf{y}) = \prod_{t=1}^T q(y_t \mid y_1, \dots, y_{t-1})$$

We then generate y_1 according to $q(\cdot \mid \text{bos})$, y_2 according to $q(\cdot \mid \text{bos}, y_1)$



How do we choose y at each step?

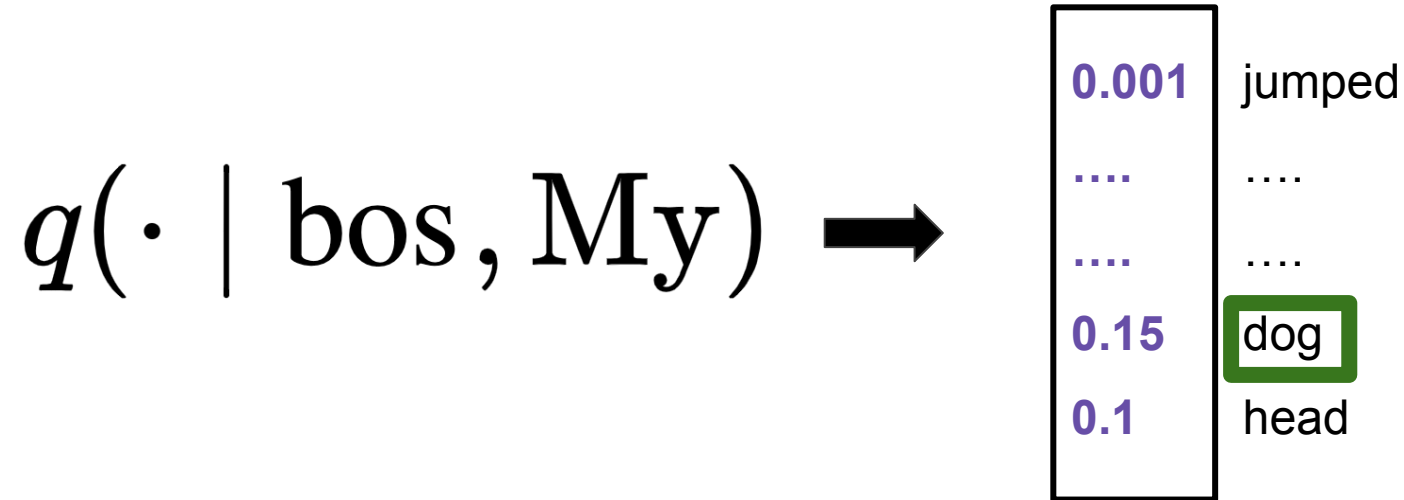
Decoding Strategy Example: Greedy Search



Greedy search says choose select the argmax at each time step:

$$y_t = \operatorname{argmax}_{y \in \bar{\mathcal{V}}} q(y \mid \mathbf{y}_{<t})$$

Decoding Strategy Example: Greedy Search



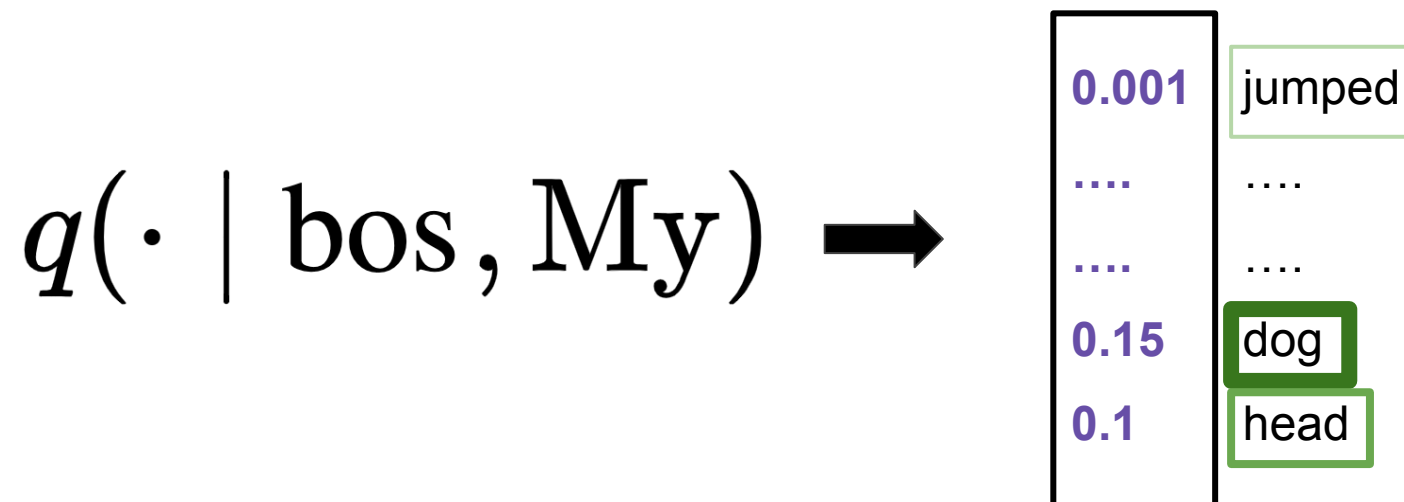
What can go wrong?

- Often leads to dull or generic text
- Prone to repetitive loops!

Greedy search says choose select the argmax at each time step:

$$y_t = \operatorname{argmax}_{y \in \bar{\mathcal{V}}} q(y \mid \mathbf{y}_{<t})$$

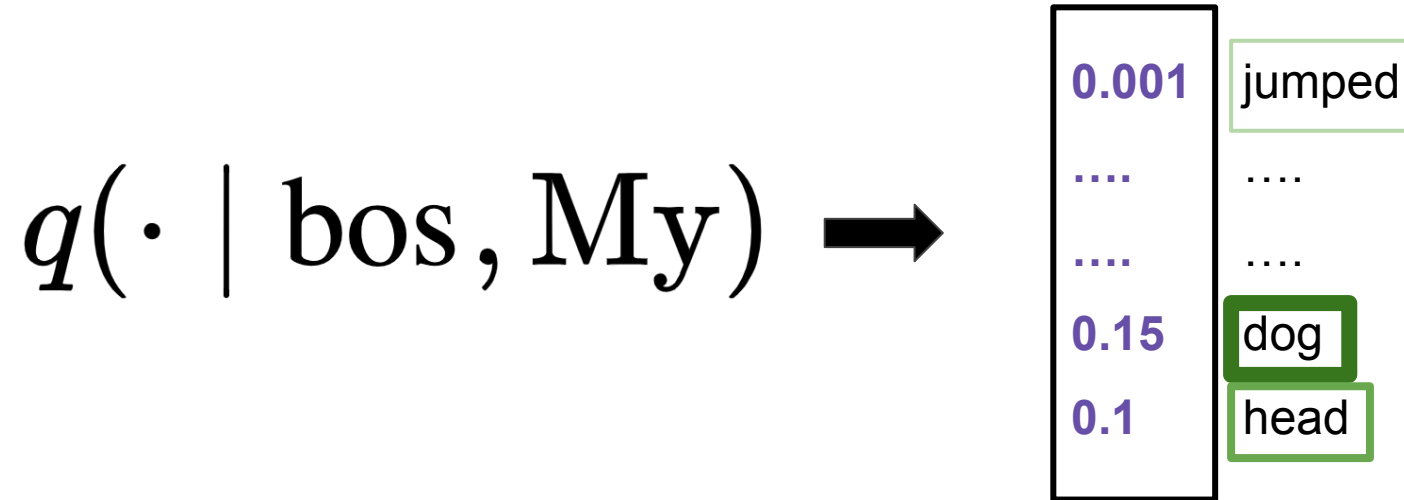
Decoding Strategy Example: Ancestral Sampling



Ancestral sampling says sample according to q at each time step:

$$y_t \sim q(\cdot \mid \mathbf{y}_{<t})$$

Decoding Strategy Example: Ancestral Sampling



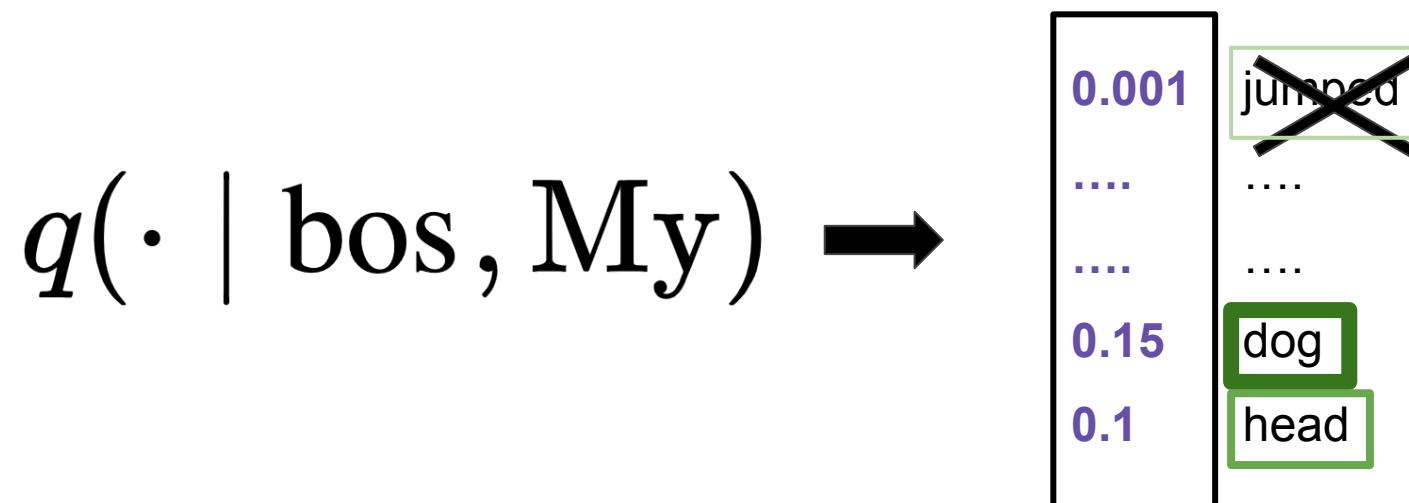
What can go wrong?

- *We often sample from the tail of the distribution, which leads to text that is not relevant or nonsensical*

Ancestral sampling says sample according to q at each time step:

$$y_t \sim q(\cdot \mid \mathbf{y}_{<t})$$

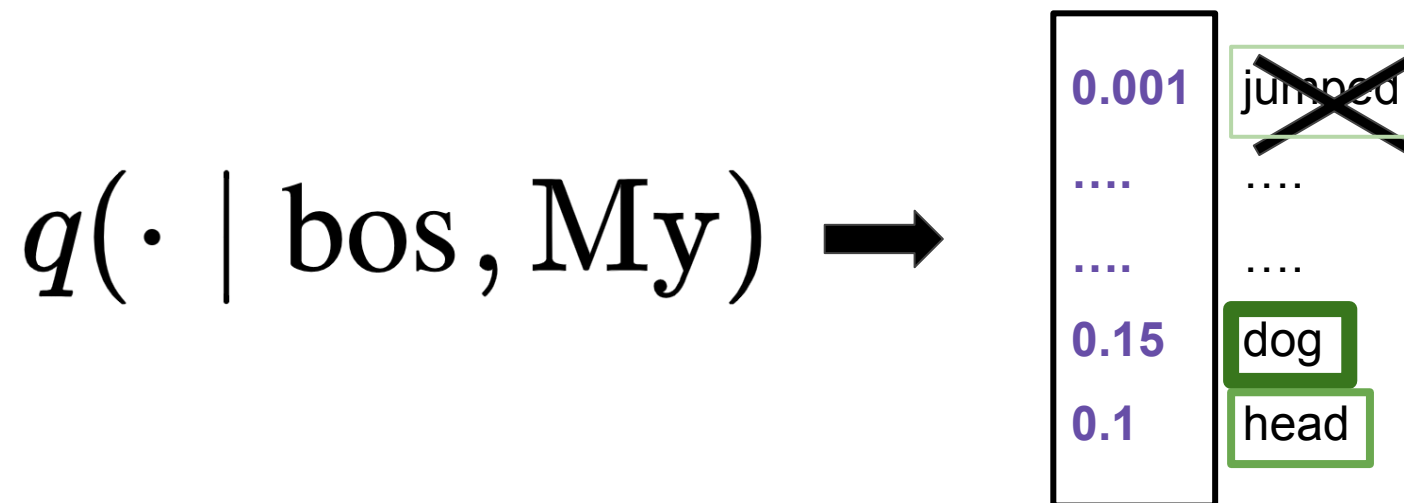
Decoding Strategy Example: Top- k Sampling



Top- k sampling says sample one of the top k of q at each time step:

$$y_t \sim \begin{cases} q(\cdot \mid \mathbf{y}_{<t}) & \mathbf{if } y_t \in \text{top}_k(q(\cdot \mid \mathbf{y}_{<t})) \\ 0 & \mathbf{otherwise} \end{cases}$$

Decoding Strategy Example: Top- k Sampling



What can go wrong?

- *The generated text is higher quality, but we still occasionally observe degenerate behavior, e.g., repetitive loops*

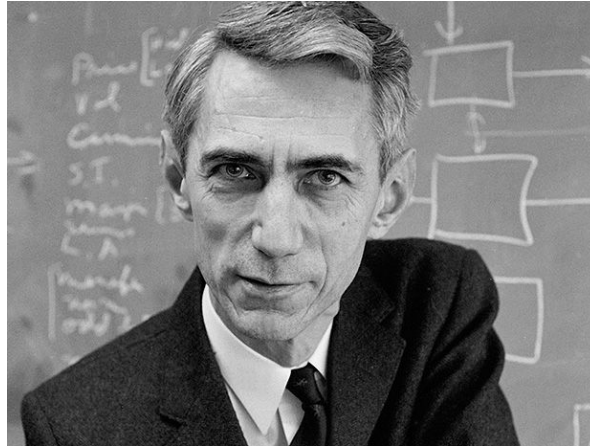
Top- k sampling says sample one of the top k of q at each time step:

$$y_t \sim \begin{cases} q(\cdot \mid \mathbf{y}_{<t}) & \text{if } y_t \in \text{top}_k(q(\cdot \mid \mathbf{y}_{<t})) \\ 0 & \text{otherwise} \end{cases}$$

An Information-Theoretic View of Language

An Information-Theoretic View of Language

Natural language is the primary means for human communication



Information theory is the mathematical study of communication

1. Can information theory help us determine when automatically generated text is human-like?
2. Can we use information-theoretic concepts to generate more human-like text?

An Information-Theoretic View of Language

The process of communicating through natural language can be interpreted as the transmission of a message via a communication channel

An Information-Theoretic View of Language

The process of communicating through natural language can be interpreted as the transmission of a message via a communication channel

Information theory suggests *two principles* that guide what makes a good sentence:

- **Principle 1:** Information should be transmitted efficiently
- **Principle 2:** Sentences should be chosen to avoid miscommunication

An Information-Theoretic View of Language

The process of communicating through natural language can be interpreted as the transmission of a message via a communication channel

Information theory suggests *two principles* that guide what makes a good sentence:

- **Principle 1:** Keep the sentence short and information dense
- **Principle 2:** Avoid moments of high information, which are hard to process

**Rephrased more
colloquially**

An Information-Theoretic View of Language

The process of communicating through natural language can be interpreted as the transmission of a message via a communication channel

Information theory suggests *two principles* that guide what makes a good sentence:

- **Principle 1:** Keep the sentence short and information dense
- **Principle 2:** Avoid moments of high information, which are hard to process

These two principles trade off!

An Information-Theoretic View of Language

The process of communicating through natural language can be interpreted as the transmission of a message via a communication channel

Information theory suggests *two principles* that guide what makes a good sentence:

- **Principle 1:** Keep the sentence short and information dense
- **Principle 2:** Avoid moments of high information, which are hard to process

These two principles trade off!

Solution: A natural solution to the above trade-off is for an algorithm to choose sentences that are around the *average information content*. Intuitively, such sentences should be informative enough, but also avoid stretches of high information.

What's Special About Average Information?

- The average information content of a distribution goes by the **entropy**
- In the case of probabilistic language generators of the form

$$q(\mathbf{y}) = \prod_{t=1}^T q(y_t \mid y_1, \dots, y_{t-1})$$

it is most natural to talk about time-step dependent entropy

- In symbols, entropy (average information) at time step t is denoted as

$$H(q(\cdot \mid \mathbf{y}_{<t})) = \sum_{y \in \bar{\mathcal{V}}} q(y \mid \mathbf{y}_{<t}) [-\log q(y \mid \mathbf{y}_{<t})]$$

**conditional entropy**

**information content**



The Expected Information Hypothesis (Meister et al. 2022a)

PS5-2: Generation, Tuesday 15:15-16:15 (Forum)

The Expected Information Hypothesis in a Picture



The Expected Information Hypothesis in a Picture



The Expected Information Hypothesis in a Picture



The Expected Information Hypothesis

Expected Information Hypothesis. Every word in a generated sentence should have an information content close to the conditional entropy of the distribution over words given prior context. That is, there exists an ε such that

$$\left| \mathbf{H}(q(\cdot \mid \mathbf{y}_{<t})) + \log q(y_t) \right| < \varepsilon$$

for every token y_t in sentence \mathbf{y} with high probability.

A Useful Definition: Local Typicality

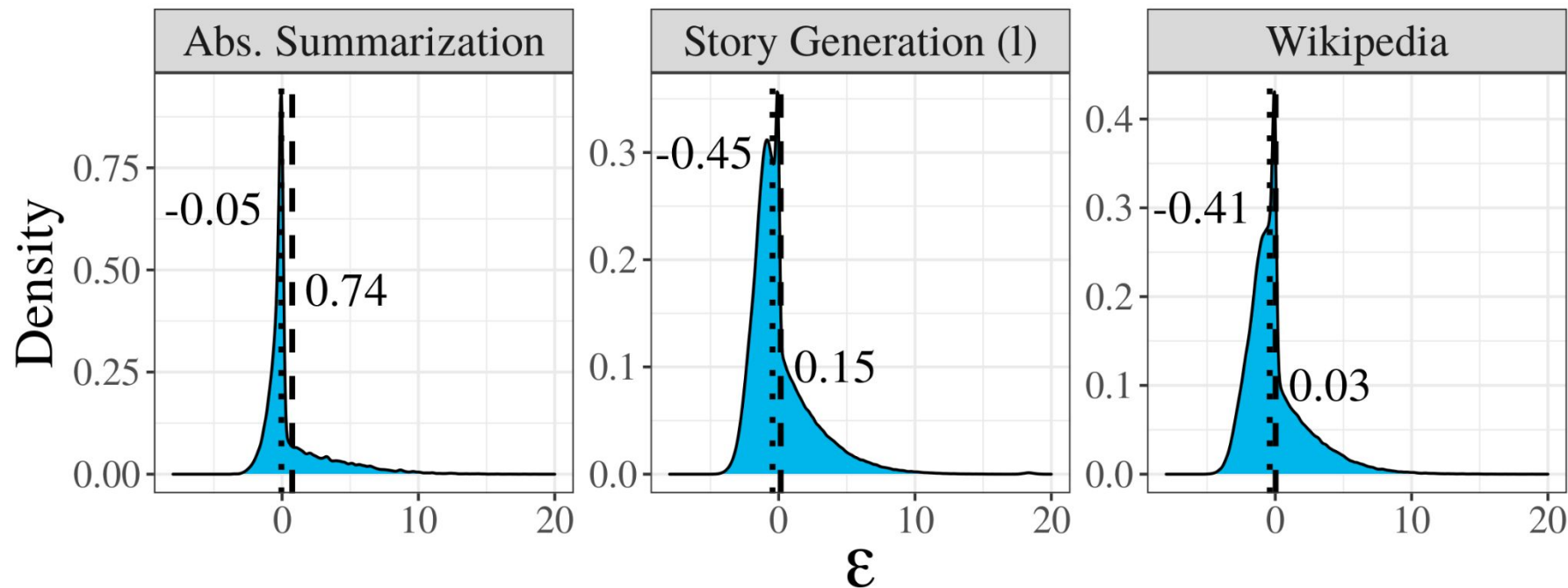
- So what *set* should speakers' utterances fall in? Let's define one!
- We define the ***locally typical set*** of the distribution q as follows

$$\mathcal{T}_\varepsilon(q) = \left\{ y : \left| \underbrace{\mathbf{H}(q)}_{\text{entropy}} - \underbrace{-\log q(y)}_{\text{information content}} \right| < \varepsilon \right\}$$

This set is defined as those y whose information content has distance of less than ε from the entropy of the distribution q

- We have a free parameter ε that we get to choose!
- **Caveat:** This is not the standard definition of typicality that you will find in information theory, e.g. Cover and Thomas (2006). It is related, though.

Empirical Evidence for the Hypothesis



The per-token distribution of the deviation (ϵ) of information content from conditional entropy on human text. The true probabilities and entropies are approximated using probabilistic models trained on the data for each task. Labels and lines indicate the mean and median deviations

Still curious about expected information hypothesis?

- See Meister et al. (2022) at *this conference!*



- We have many, many experiments that support the hypothesis

On the probability–quality paradox in language generation

Clara Meister 🤖 Gian Wiher 🤖 Tiago Pimentel 🤖 Ryan Cotterell 🤖
ETH Zürich University of Cambridge
clara.meister@inf.ethz.ch gian.wiher@inf.ethz.ch
tp472@cam.ac.uk ryan.cotterell@inf.ethz.ch

Abstract

When generating natural language from neural probabilistic models, high probability does not always coincide with high quality: It

appears to have an inflection point,² i.e., quality and probability are positively correlated only until a certain threshold, after which the correlation becomes negative. While the existence of such a trend has received informal explanations (see, e.g.,

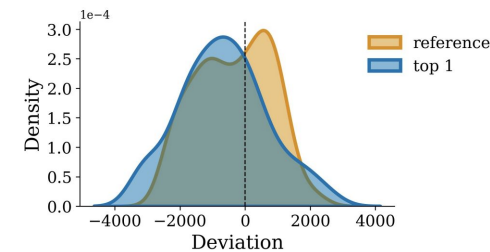


Figure 2: The distribution of the difference in total information content for (1) test-set references and (2) top-ranked model-generated strings from the (conditional) entropy of the model from which they were generated.

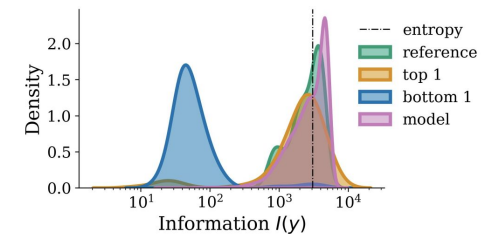
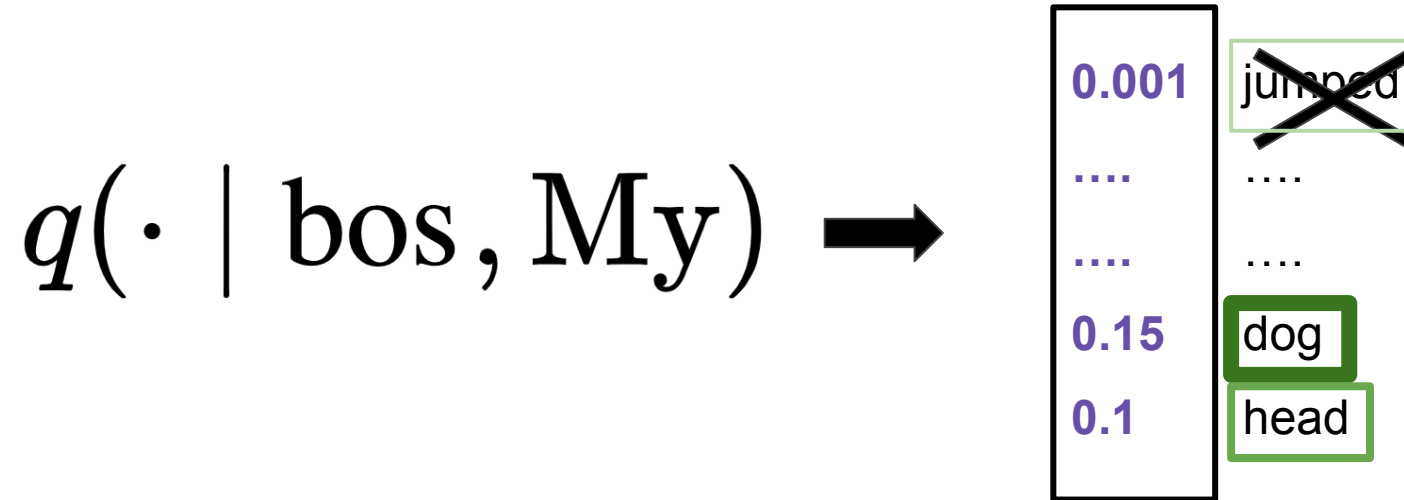


Figure 1: The distribution over information $I(y)$ values of: MODEL, the model, as estimated using samples from q ; REFERENCE, the reference strings; TOP 1 and BOTTOM 1, model-generated strings ranked first and last (respectively) among all decoding strategies by human annotators. The latter 3 are all w.r.t. a held-out test set. Same graph is reproduced for individual decoding strategies in App. B.



Typical sampling: From hypothesis to algorithm (Meister et al. 2022b)

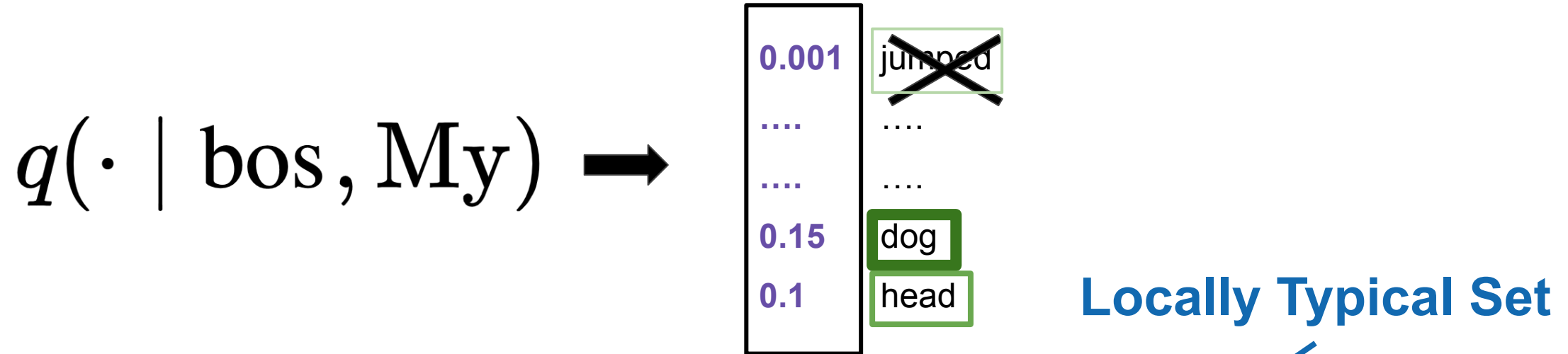
A New Decoding Strategy: Typical Sampling



Typical sampling says sample from the locally typical set at each time step:

$$y_t \sim \begin{cases} q(\cdot \mid \mathbf{y}_{<t}) & \text{if } y_t \in \mathcal{T}_\varepsilon(q(\cdot \mid \mathbf{y}_{<t})) \\ 0 & \text{otherwise} \end{cases}$$

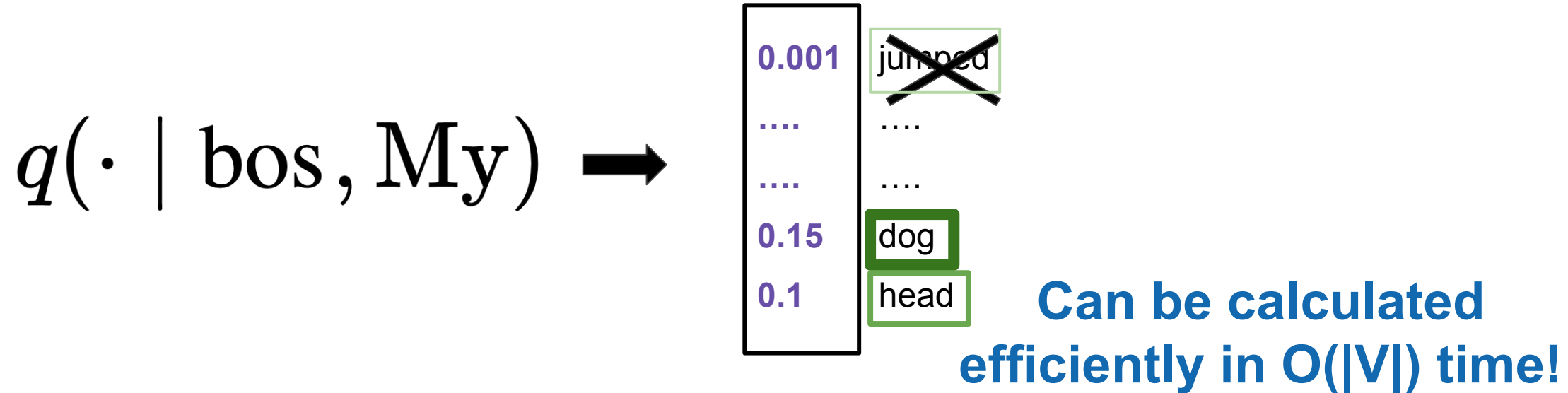
A New Decoding Strategy: Typical Sampling



Typical sampling says sample from the locally typical set at each time step:

$$y_t \sim \begin{cases} q(\cdot \mid \mathbf{y}_{<t}) & \text{if } y_t \in \mathcal{T}_\varepsilon(q(\cdot \mid \mathbf{y}_{<t})) \\ 0 & \text{otherwise} \end{cases}$$

A New Decoding Strategy: Typical Sampling

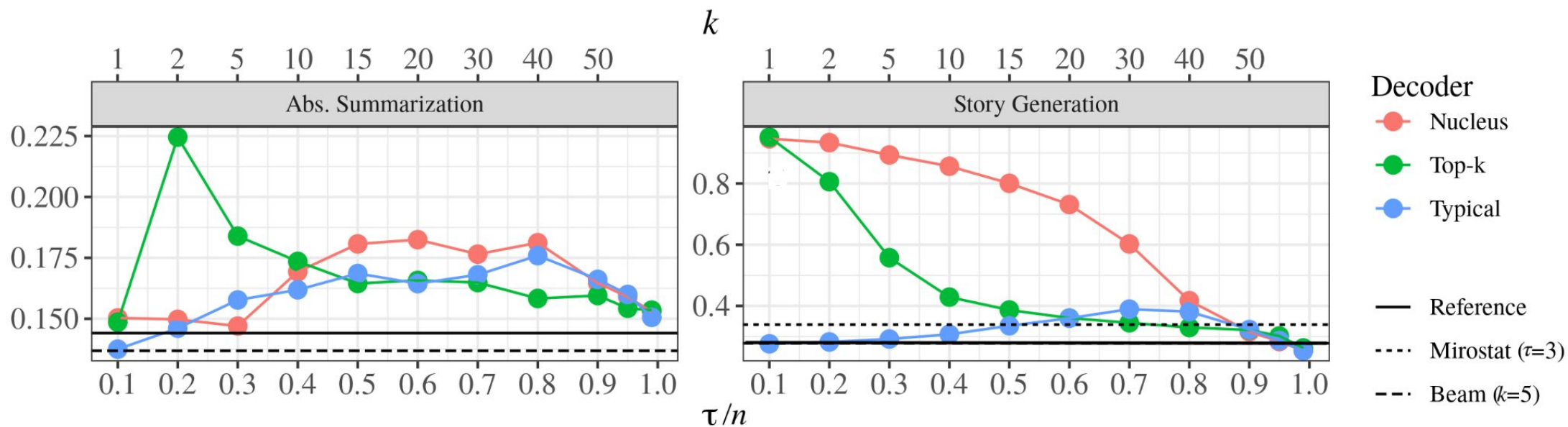


Typical sampling says sample from the locally typical set at each time step:

$$y_t \sim \begin{cases} q(\cdot \mid \mathbf{y}_{<t}) & \text{if } y_t \in \mathcal{T}_\varepsilon(q(\cdot \mid \mathbf{y}_{<t})) \\ 0 & \text{otherwise} \end{cases}$$

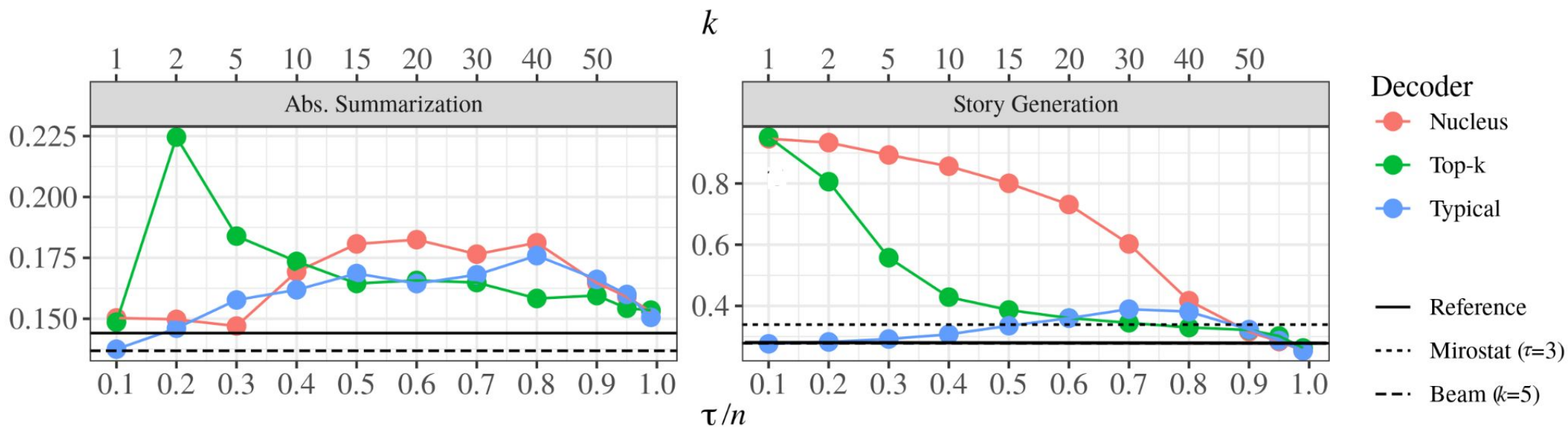
How well does it compare? nUmBeRs wEnt uP

	Abstractive Summarization					Story Generation				
	PPL	REP	Zipf	MAUVE	Human	PPL	REP	Zipf	MAUVE	Human
Reference	11.51	0.13	0.76	-	4.07	20.06	0.28	1.09	-	4.23
Nucleus	3.28	0.16	0.93	0.93	3.97	9.65	0.32	1.25	0.95	4.00
Top- k	3.12	0.16	0.93	0.96	3.97	7.77	0.34	1.42	0.97	4.08
Typical	4.12	0.15	0.92	0.98	4.03	17.32	0.28	1.26	0.92	4.10



How well does it compare? nUmBeRs wEnt uP

	Abstractive Summarization					Story Generation				
	PPL	REP	Zipf	MAUVE	Human	PPL	REP	Zipf	MAUVE	Human
Reference	11.51	0.13	0.76	-	4.07	20.06	0.28	1.09	-	4.23
Nucleus	3.28	0.16	0.93	0.93	3.97	9.65	0.32	1.25	0.95	4.00
Top- k	3.12	0.16	0.93	0.96	3.97	7.77	0.34	1.42	0.97	4.08
Typical	4.12	0.15	0.92	0.98	4.03	17.32	0.28	1.26	0.92	4.10



How well does it compare? Qualitative results

Prompt	(CNN) The attorney for a suburban New York cardiologist charged in what authorities say was a failed scheme to have another physician hurt or killed is calling the allegations against his client “completely unsubstantiated...”
Reference	A lawyer for Dr. Anthony Moschetto says the charges against him are baseless. Moschetto, 54, was arrested for selling drugs and weapons, prosecutors say. Authorities allege Moschetto hired accomplices to burn down the practice of former associate.
Beam Search	Dr. Anthony Moschetto faces criminal solicitation, conspiracy, burglary, arson and weapons charges. “None of anything in this case has any evidentiary value,” his attorney says.
Nucleus Sampling	Dr. Anthony Moschetto, 54, pleaded not guilty to charges Wednesday. Two men -- identified as James Chmela and James Kalamaras -- were named as accomplices.
Top-<i>k</i> Sampling	Dr. Anthony Moschetto is accused of providing police with weapons and prescription drugs. Authorities say he was part of a conspiracy to harm or kill a rival doctor. His attorney calls the allegations against his client “completely unsubstantiated”
Typical Sampling	Dr. Anthony Moschetto is charged with crimes including arson, conspiracy, burglary, prescription sale, weapons charges. His attorney says “none of anything in this case has any evidentiary value”


How well does it compare? Qualitative results


Prompt	(CNN) The attorney for a suburban New York cardiologist charged in what authorities say was a failed scheme to have another physician hurt or killed is calling the allegations against his client “completely unsubstantiated...”
Reference	A lawyer for Dr. Anthony Moschetto says the charges against him are baseless. Moschetto, 54, was arrested for selling drugs and weapons, prosecutors say. Authorities allege Moschetto hired accomplices to burn down the practice of former associate.
<i>Beam Search</i>	Dr. Anthony Moschetto faces criminal solicitation, conspiracy, burglary, arson and weapons charges. “None of anything in this case has any evidentiary value,” his attorney says.
Nucleus Sampling	Dr. Anthony Moschetto, 54, pleaded not guilty to charges Wednesday. Two men -- identified as James Chmela and James Kalamaras -- were named as accomplices.
Top-k Sampling	Dr. Anthony Moschetto is accused of providing police with weapons and prescription drugs. Authorities say he was part of a conspiracy to harm or kill a rival doctor. His attorney calls the allegations against his client “completely unsubstantiated”
<i>Typical Sampling</i>	Dr. Anthony Moschetto is charged with crimes including arson, conspiracy, burglary, prescription sale, weapons charges. His attorney says “none of anything in this case has any evidentiary value”


How well does it compare? Qualitative results

<p>Prompt</p>	<p>(CNN) The attorney for a suburban New York doctor charged in what authorities say was a failed scheme to have another physician hurt or killed. Investigations against his client “completely unsubstantiated...”</p>
<p>Reference</p>	<p>A lawyer for Dr. Anthony Moschetto says he was arrested for selling drugs and weapons to accomplices to burn down the practice. Moschetto, 54, was charged with arson, conspiracy, burglary, weapons charges. His attorney says “none of anything in this case has any evidentiary value”</p>
<p><i>Beam Search</i></p>	<p>Dr. Anthony Moschetto faces criminal charges. “None of anything in this case has any evidentiary value”</p>
<p>Nucleus Sampling</p>	<p>Dr. Anthony Moschetto, 54, pleaded not guilty to charges of providing police with weapons. Authorities say he was part of a conspiracy to harm or kill a rival doctor. Chmela and James Kalamaras -- were named as accomplices in the plot. Moschetto hired</p>
<p>Top-k Sampling</p>	<p>Dr. Anthony Moschetto is accused of providing police with weapons. Authorities say he was part of a conspiracy to harm or kill a rival doctor. Investigations against his client “completely unsubstantiated”</p>
<p><i>Typical Sampling</i></p>	<p>Dr. Anthony Moschetto is charged with crimes including arson, conspiracy, burglary, weapons sale, weapons charges. His attorney says “none of anything in this case has any evidentiary value”</p>

If Beam Search is the Answer, What was the Question?

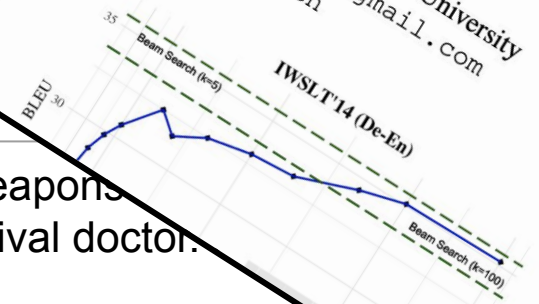
Clara Meister  University of Cambridge
 clara.meister@inf.ethz.ch

Ryan Cotterell  Johns Hopkins University
 ryan.cotterell@inf.ethz.ch

Tim Vieira  Johns Hopkins University
 tim.vieira@gmail.com

Abstract

Quite surprisingly, exact maximum a posteriori (MAP) decoding of neural language generators frequently leads to low-quality results (Stahlberg and Byrne, 2019). Rather, most state-of-the-art results on language generation



Try out typical sampling!



Typical sampling is implemented in the Hugging Face transformers library!



HUGGING FACE

```
241 class TypicalLogitsWarper(LogitsWarper):
242     def __init__(self, mass: float = 0.9, filter_value: float = -float("Inf"), min_tokens_to_keep: int = 1):
243         mass = float(mass)
244         if not (mass > 0 and mass < 1):
245             raise ValueError(f"`typical_p` has to be a float > 0 and < 1, but is {mass}")
246
247         self.filter_value = filter_value
248         self.mass = mass
249         self.min_tokens_to_keep = min_tokens_to_keep
250
251     def __call__(self, input_ids: torch.LongTensor, scores: torch.FloatTensor) -> torch.FloatTensor:
252
253         # calculate entropy
254         normalized = torch.nn.functional.log_softmax(scores, dim=-1)
255         p = torch.exp(normalized)
256         ent = -(normalized * p).nansum(-1, keepdim=True)
257
258         # shift and sort
259         shifted_scores = torch.abs((-normalized) - ent)
260         sorted_scores, sorted_indices = torch.sort(shifted_scores, descending=False)
261         sorted_logits = scores.gather(-1, sorted_indices)
262         cumulative_probs = sorted_logits.softmax(dim=-1).cumsum(dim=-1)
263
264         # Remove tokens with cumulative mass above the threshold
265         last_ind = (cumulative_probs < self.mass).sum(dim=1)
266         last_ind[last_ind < 0] = 0
267         sorted_indices_to_remove = sorted_scores > sorted_scores.gather(1, last_ind.view(-1, 1))
268         if self.min_tokens_to_keep > 1:
269             # Keep at least min_tokens_to_keep (set to min_tokens_to_keep-1 because we add the first one below)
270             sorted_indices_to_remove[..., : self.min_tokens_to_keep] = 0
271         indices_to_remove = sorted_indices_to_remove.scatter(1, sorted_indices, sorted_indices_to_remove)
272
273         scores = scores.masked_fill(indices_to_remove, self.filter_value)
274         return scores
275
```


Fin