COMP 3361 Natural Language Processing

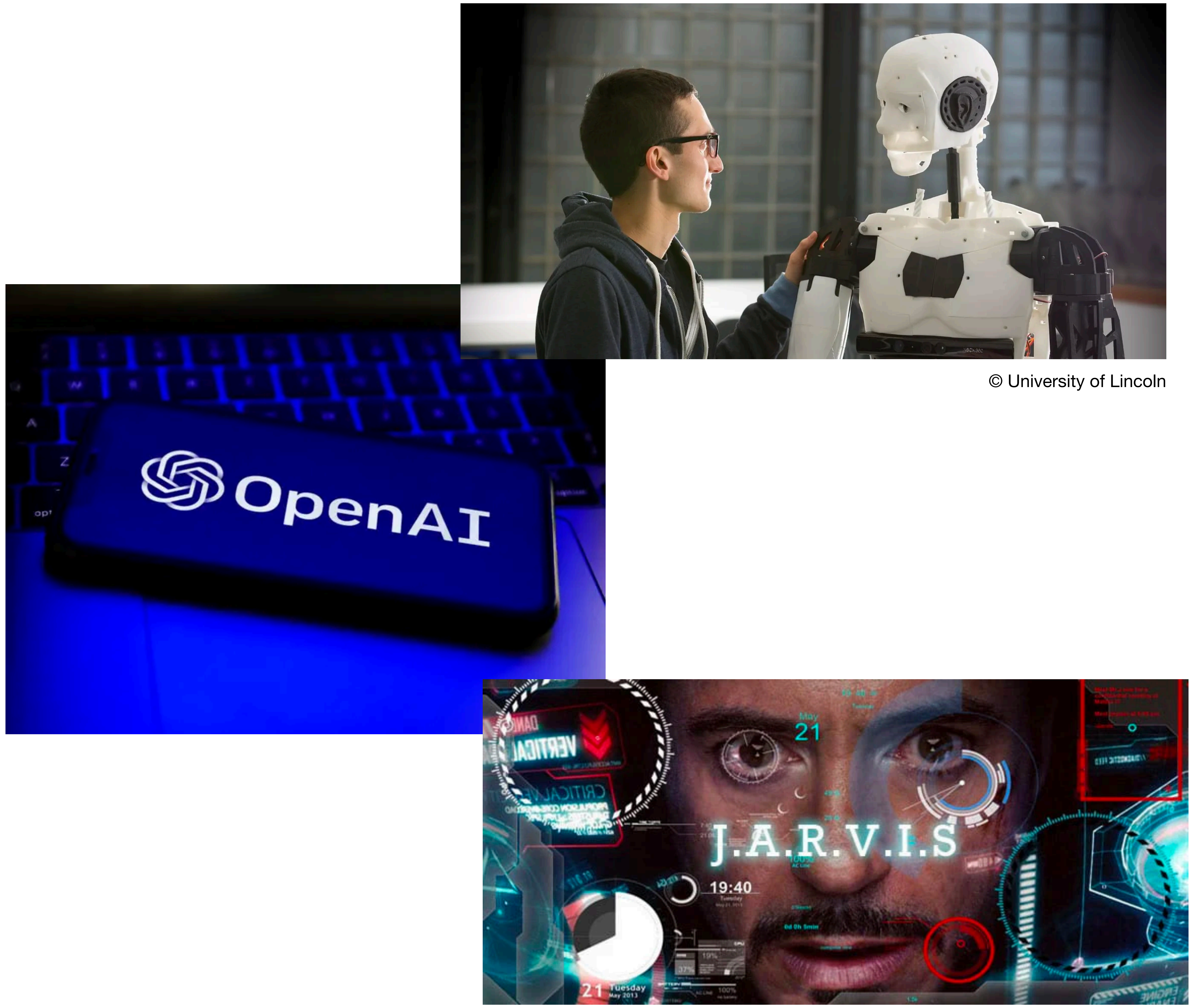
Lecture 14: Natural language generation with LLMs

Spring 2024

Many materials from CSE447@UW (Jaehun Jung) with special thanks!

Natural language generation with LLMs

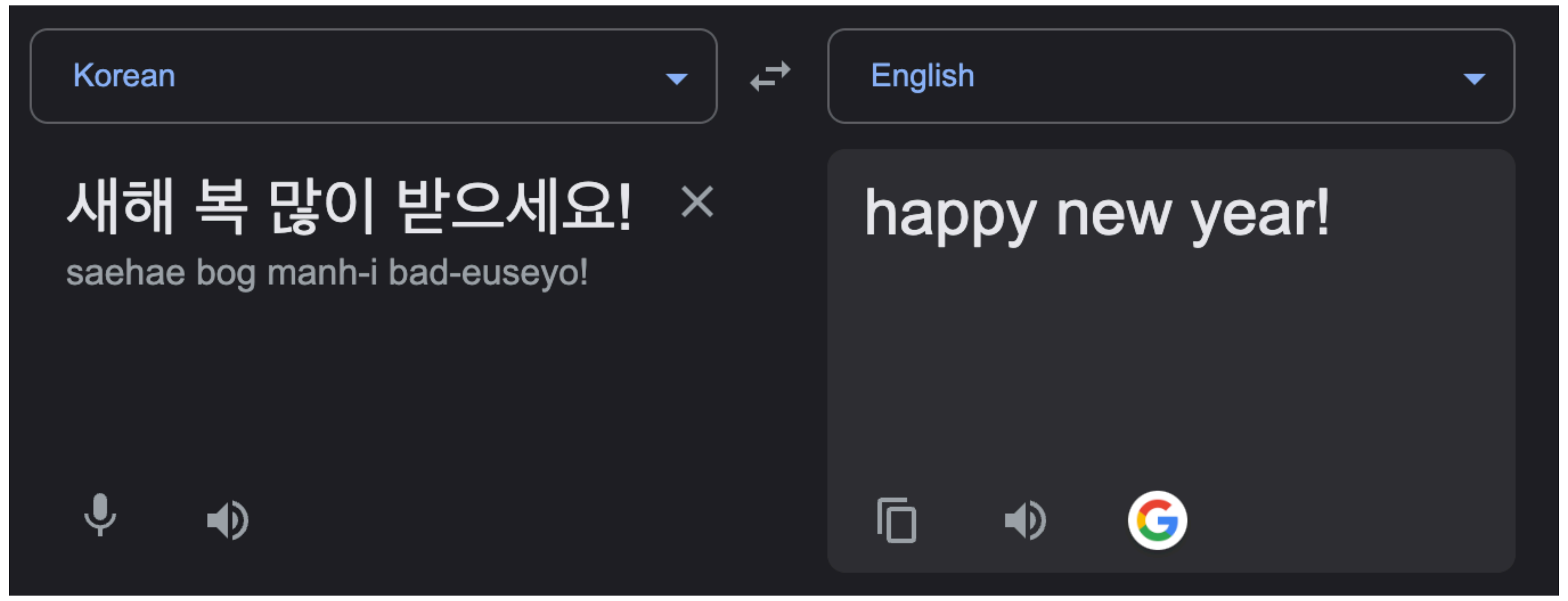
- Large Language Models are (mostly) **natural language generation (NLG)** systems!
- The process of generating natural language test with LLMs is called **decoding.**
- LLMs could perform NLG tasks (e.g., see next slides), and many **other tasks** (e.g., question answering, sentiment analysis, code generation, information extraction…) can be **formatted as NLG tasks too**!



© University of Lincoln

© Marvel Studios

Machine translation



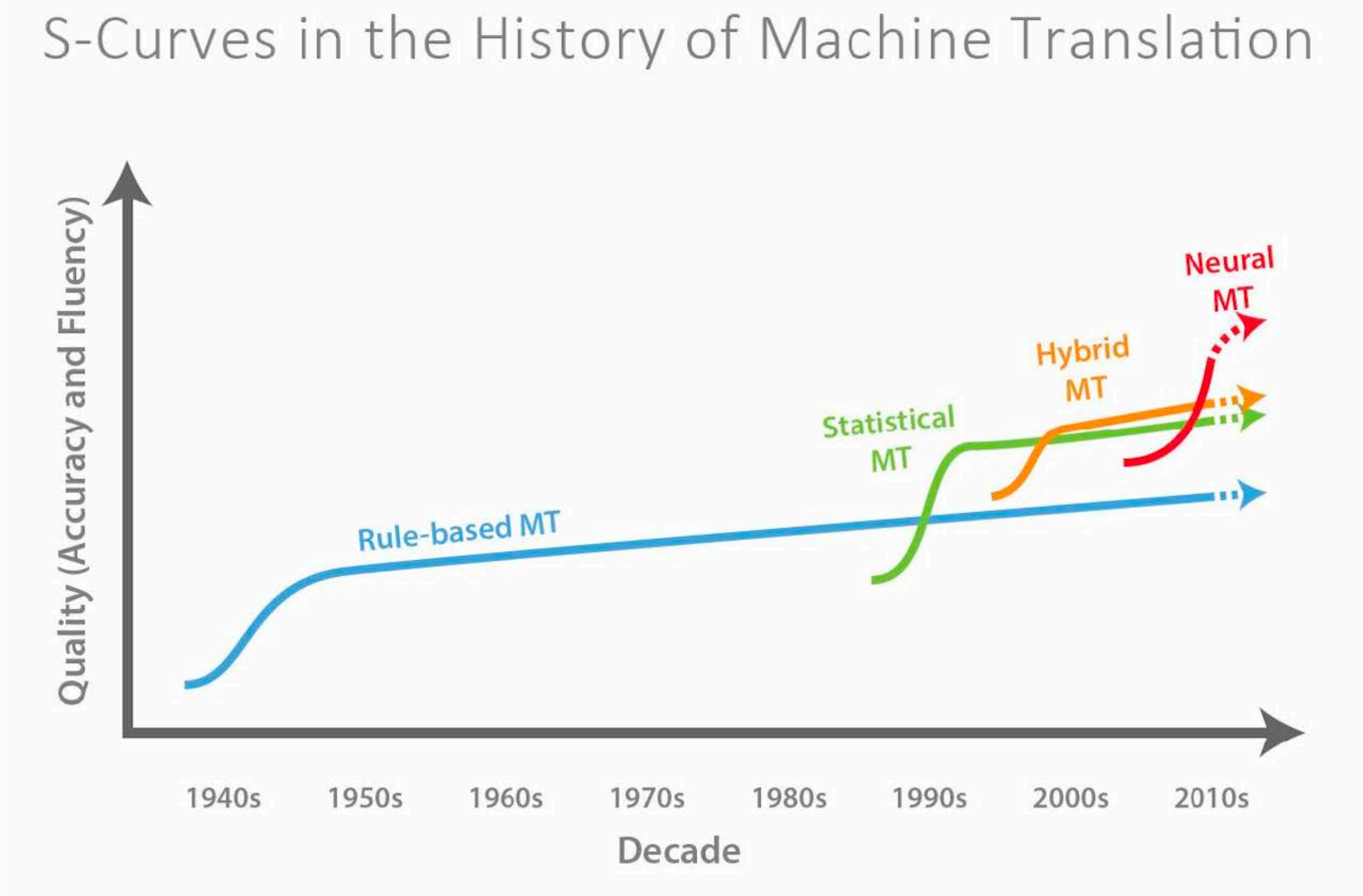
Korean

English

새해 복 많이 받으세요!

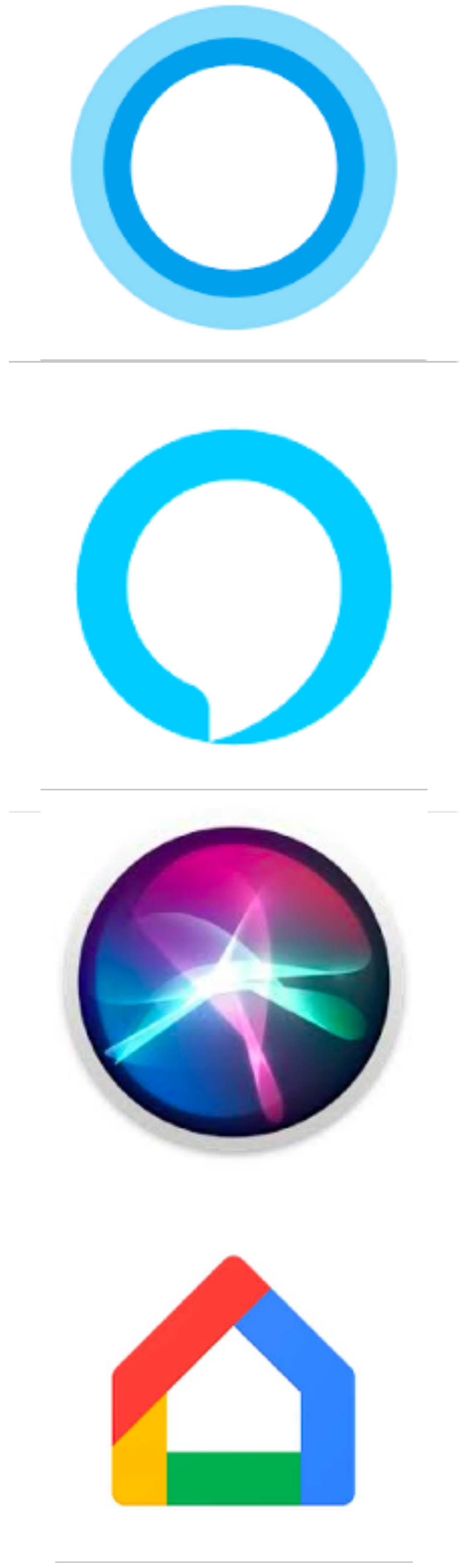
saehae bog manh-i bad-euseyo!

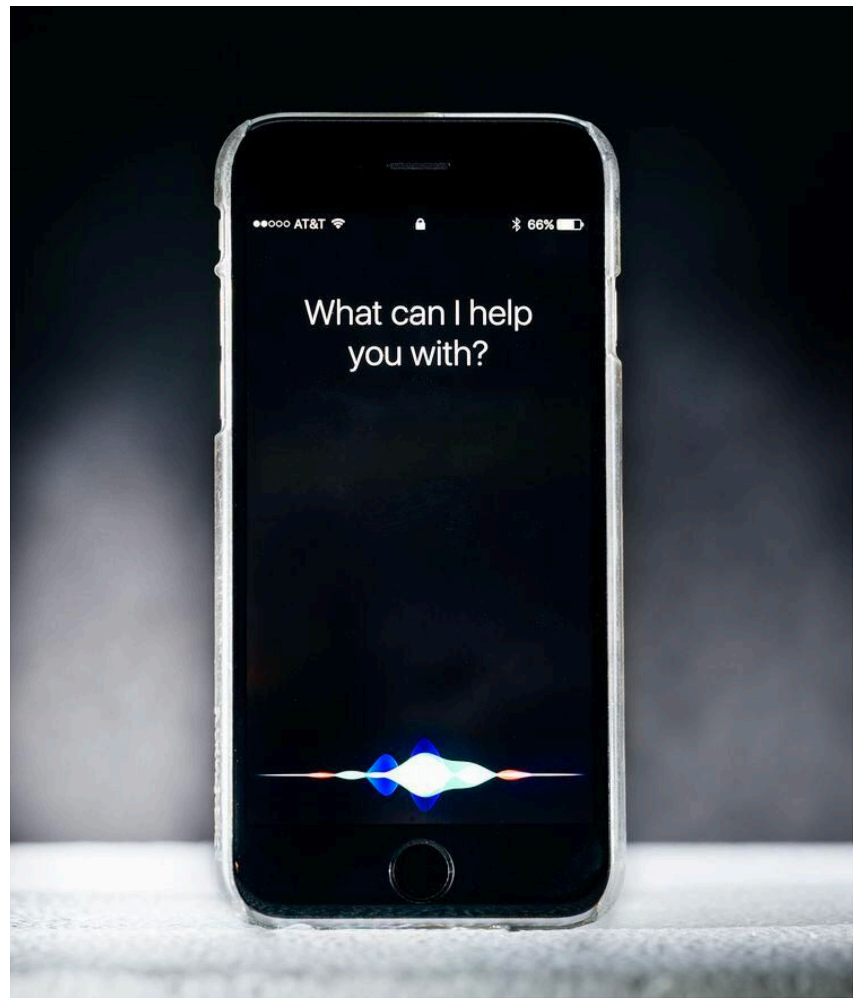
happy new year!

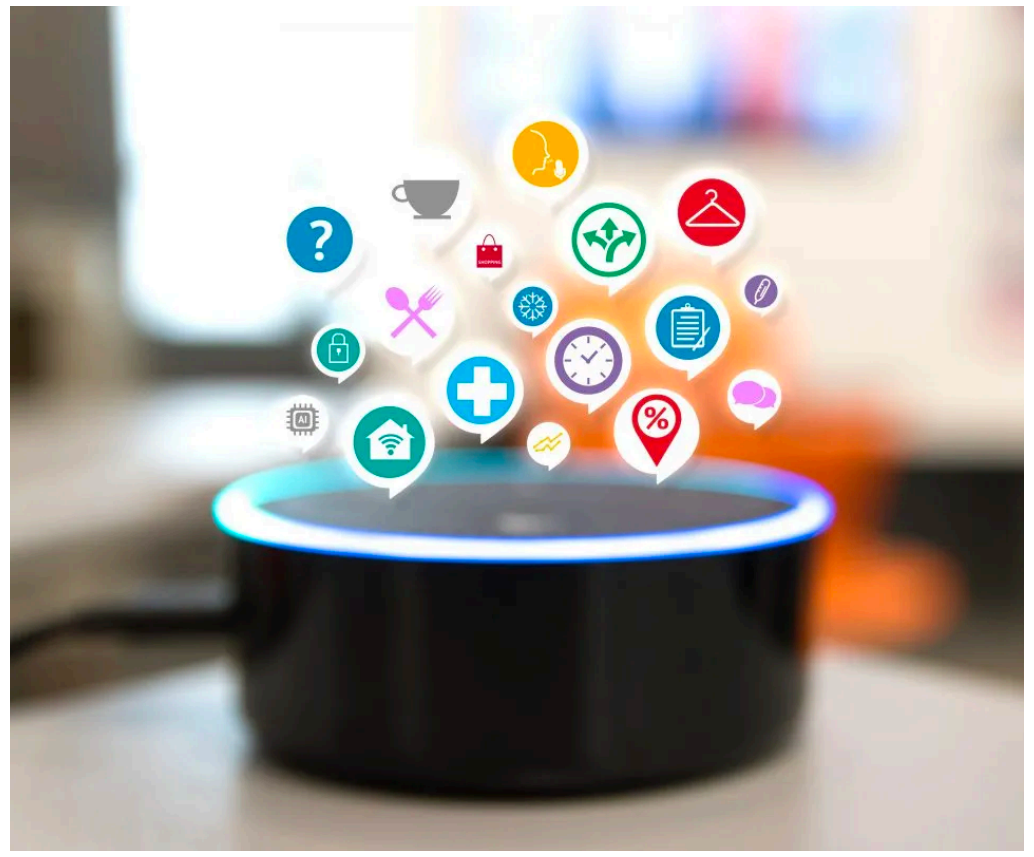


S-Curves in the History of Machine Translation

Dialogue systems

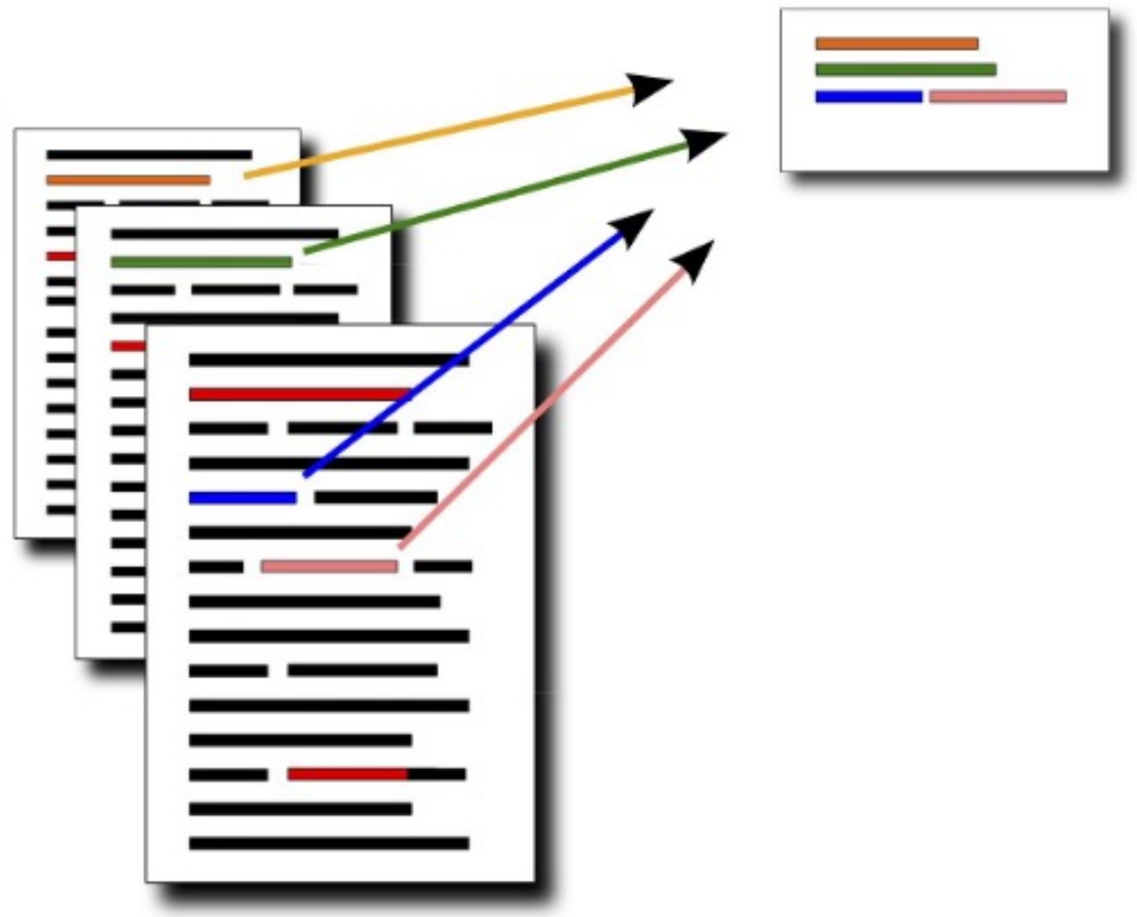






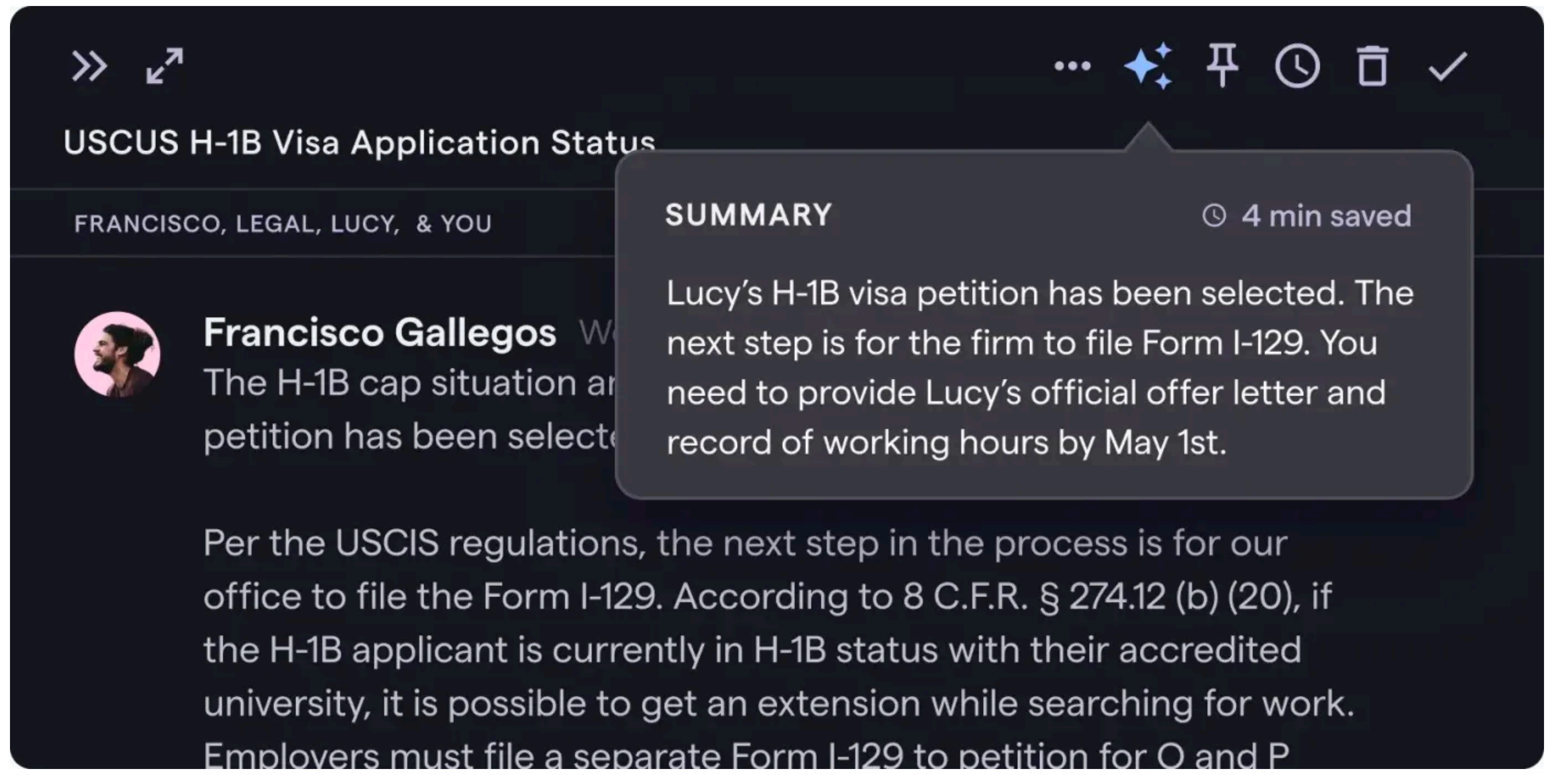
Summarization

Document Summarization



© http://mogren.one/lic/

Email Summarization



USCUS H-1B Visa Application Status

FRANCISCO, LEGAL, LUCY, & YOU

Francisco Gallegos W

The H-1B cap situation ar

petition has been select

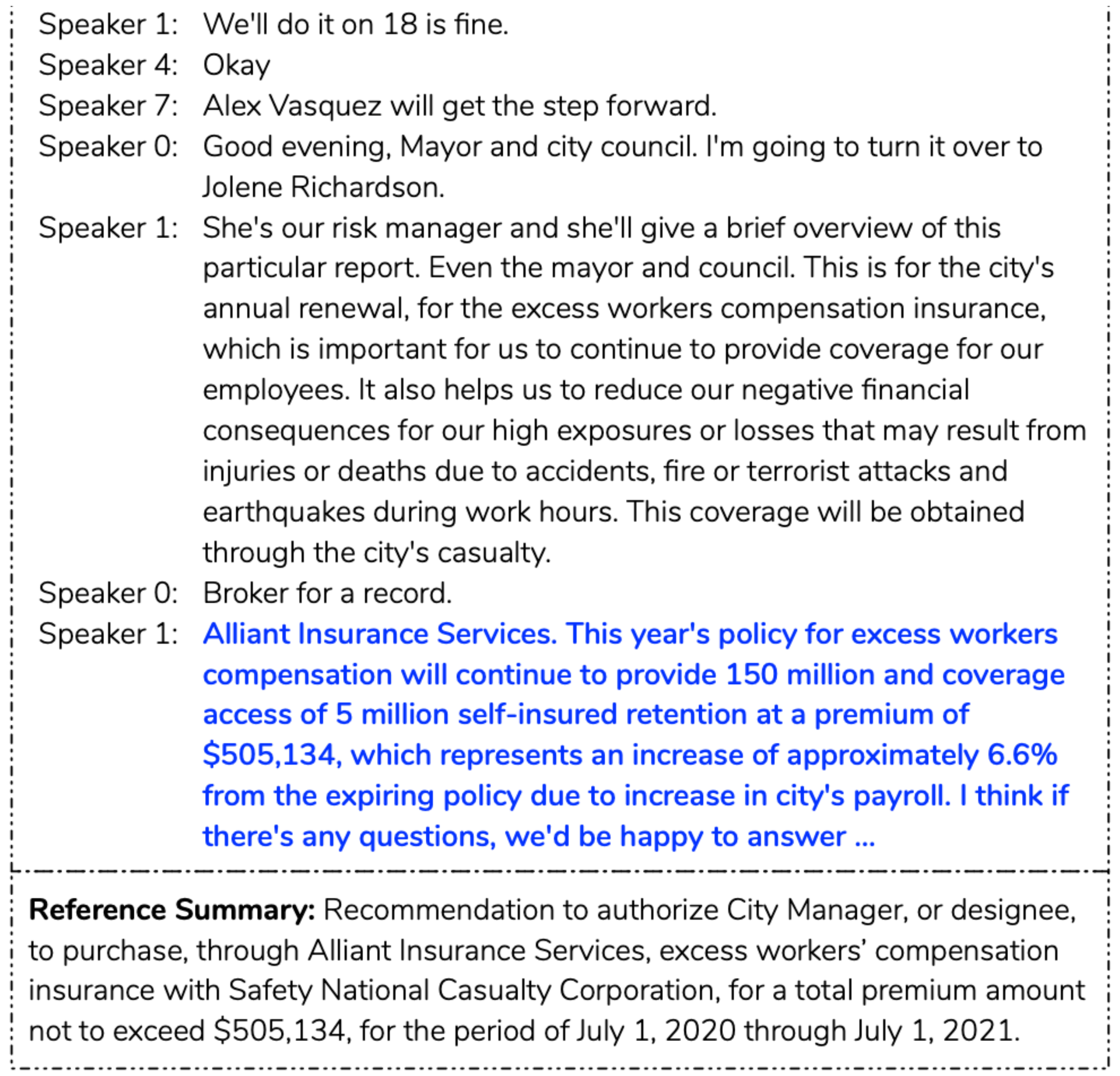
Per the USCIS regulations, the next step in the process is for our office to file the Form I-129. According to 8 C.F.R. § 274.12 (b) (20), if the H-1B applicant is currently in H-1B status with their accredited university, it is possible to get an extension while searching for work. Employers must file a separate Form I-129 to petition for O and P

SUMMARY — 4 min saved

Lucy's H-1B visa petition has been selected. The next step is for the firm to file Form I-129. You need to provide Lucy's official offer letter and record of working hours by May 1st.

© techcrunch.com

Meeting Summarization



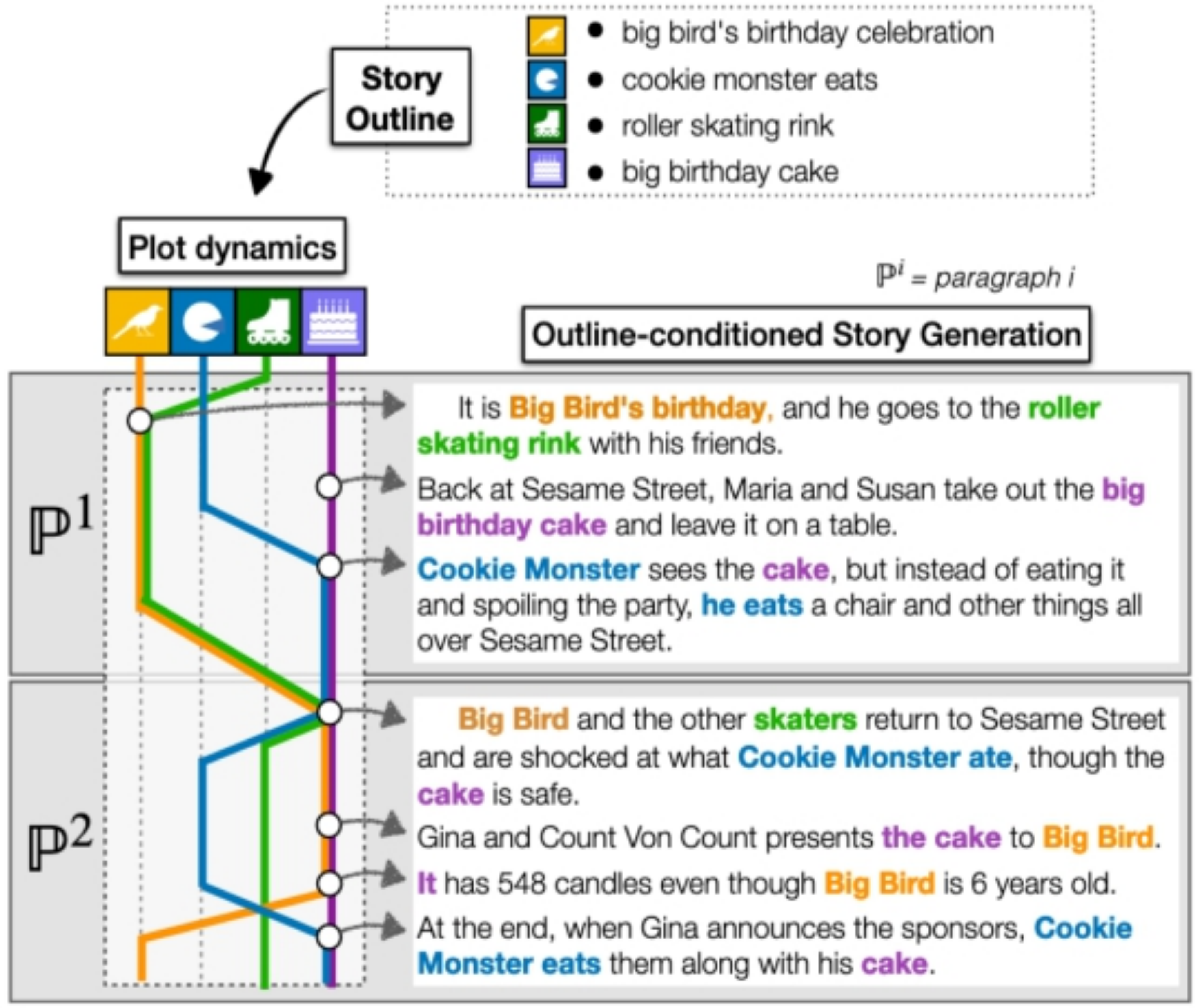
Speaker 1: We'll do it on 18 is fine.
Speaker 4: Okay
Speaker 7: Alex Vasquez will get the step forward.
Speaker 0: Good evening, Mayor and city council. I'm going to turn it over to Jolene Richardson.
Speaker 1: She's our risk manager and she'll give a brief overview of this particular report. Even the mayor and council. This is for the city's annual renewal, for the excess workers compensation insurance, which is important for us to continue to provide coverage for our employees. It also helps us to reduce our negative financial consequences for our high exposures or losses that may result from injuries or deaths due to accidents, fire or terrorist attacks and earthquakes during work hours. This coverage will be obtained through the city's casualty.
Speaker 0: Broker for a record.
Speaker 1: Alliant Insurance Services. This year's policy for excess workers compensation will continue to provide 150 million and coverage access of 5 million self-insured retention at a premium of $505,134, which represents an increase of approximately 6.6% from the expiring policy due to increase in city's payroll. I think if there's any questions, we'd be happy to answer ...

Reference Summary: Recommendation to authorize City Manager, or designee, to purchase, through Alliant Insurance Services, excess workers' compensation insurance with Safety National Casualty Corporation, for a total premium amount not to exceed $505,134, for the period of July 1, 2020 through July 1, 2021.

Hu et al., 2023

More interesting NLG uses

Creative Stories


Story Outline
Plot dynamics
$\mathbb{P}^i$ = paragraph i
Outline-conditioned Story Generation
$\mathbb{P}^1$
$\mathbb{P}^2$

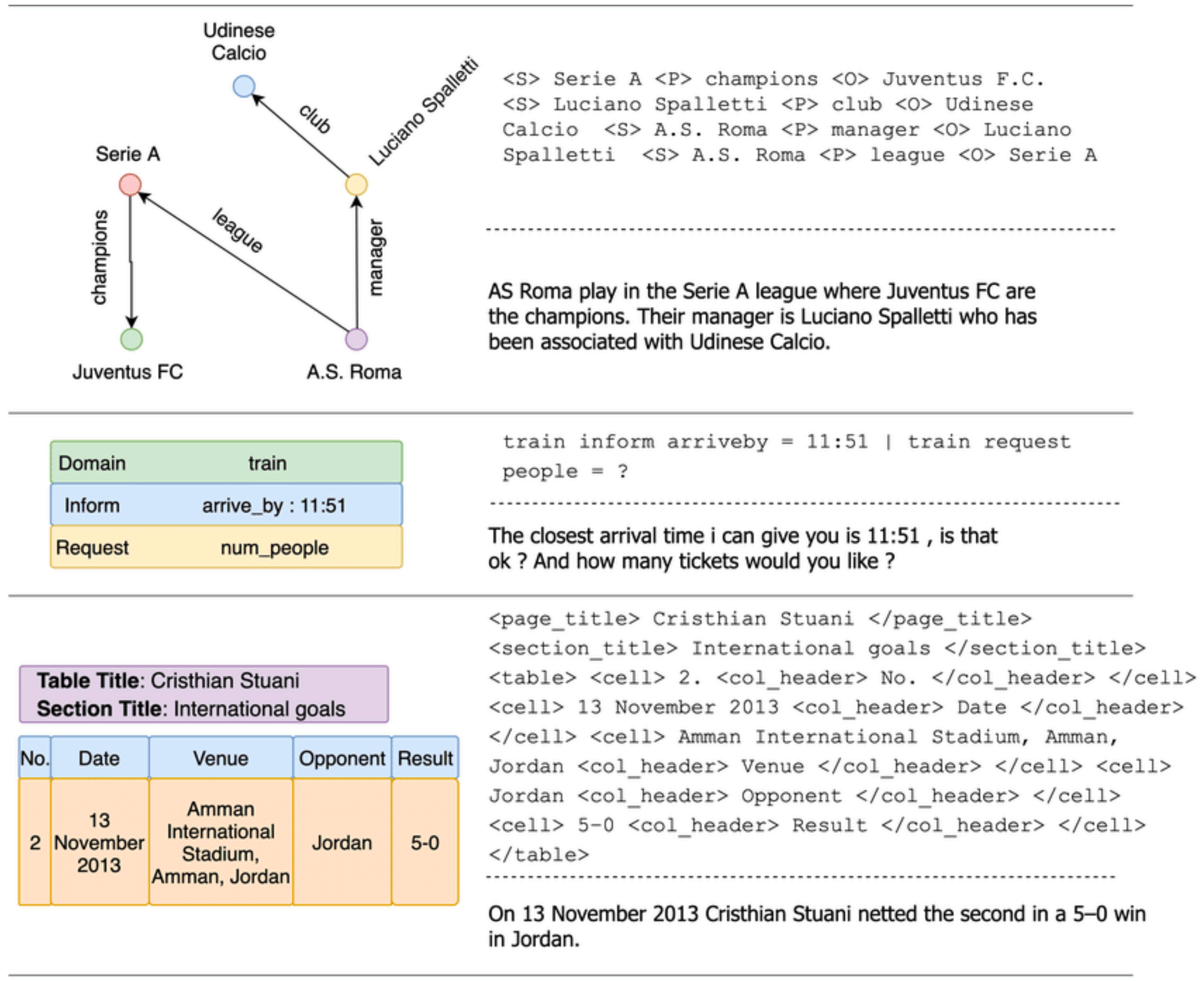
- big bird's birthday celebration
- cookie monster eats
- roller skating rink
- big birthday cake

It is Big Bird's birthday, and he goes to the roller skating rink with his friends. Back at Sesame Street, Maria and Susan take out the big birthday cake and leave it on a table. Cookie Monster sees the cake, but instead of eating it and spoiling the party, he eats a chair and other things all over Sesame Street.

Big Bird and the other skaters return to Sesame Street and are shocked at what Cookie Monster ate, though the cake is safe. Gina and Count Von Count presents the cake to Big Bird. It has 548 candles even though Big Bird is 6 years old. At the end, when Gina announces the sponsors, Cookie Monster eats them along with his cake.

Rashkin et al., EMNLP 2020

Data-to-Text Generation



```
<S> Serie A <P> champions <O> Juventus F.C.
<S> Luciano Spalletti <P> club <O> Udinese
Calcio  <S> A.S. Roma <P> manager <O> Luciano
Spalletti  <S> A.S. Roma <P> league <O> Serie A
```

AS Roma play in the Serie A league where Juventus FC are the champions. Their manager is Luciano Spalletti who has been associated with Udinese Calcio.

Domain	train
Inform	arrive_by : 11:51
Request	num_people

```
train inform arriveby = 11:51 | train request
people = ?
```

The closest arrival time i can give you is 11:51 , is that ok ? And how many tickets would you like ?

Table Title: Cristhian Stuani
Section Title: International goals

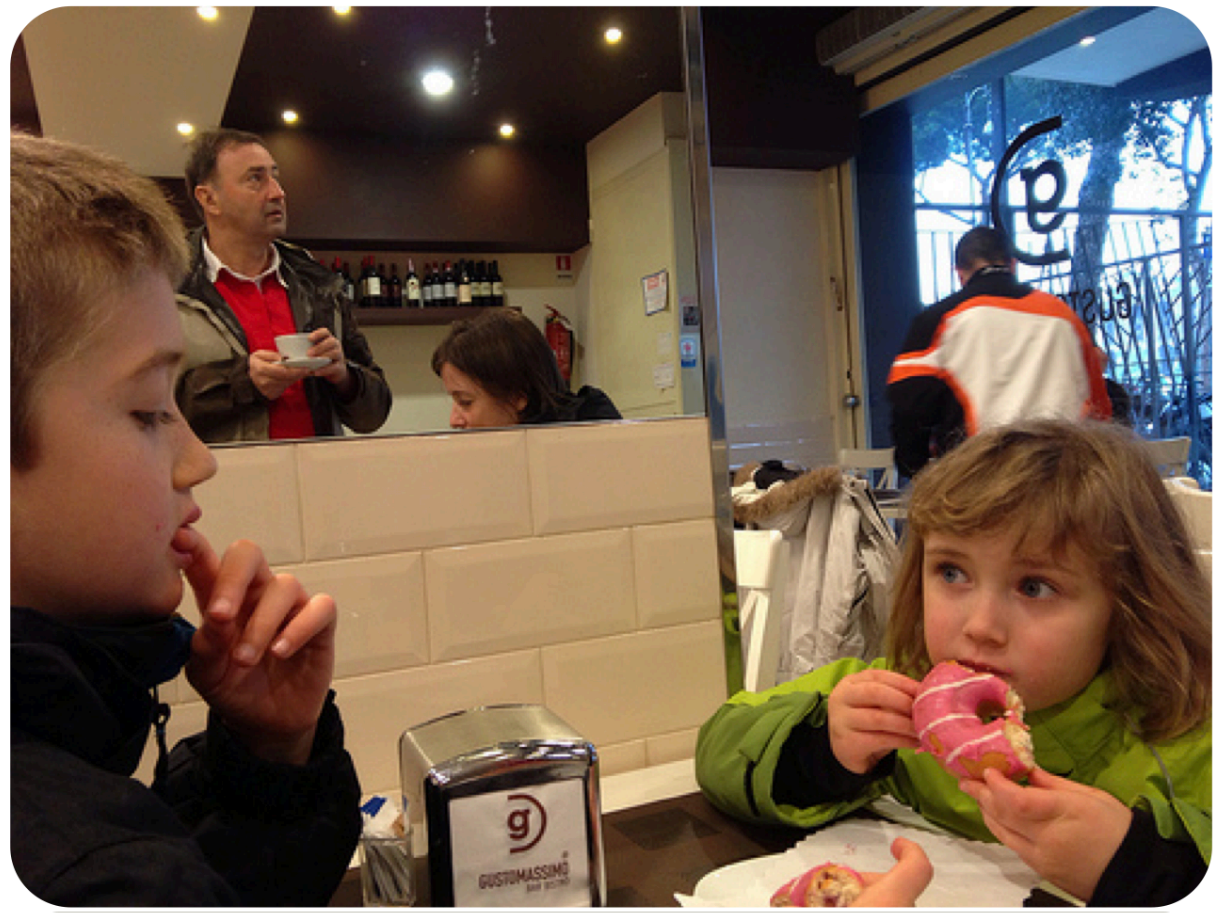
No.	Date	Venue	Opponent	Result
2	13 November 2013	Amman International Stadium, Amman, Jordan	Jordan	5-0

```
<page_title> Cristhian Stuani </page_title>
<section_title> International goals </section_title>
<table> <cell> 2. <col_header> No. </col_header> </cell>
<cell> 13 November 2013 <col_header> Date </col_header>
</cell> <cell> Amman International Stadium, Amman,
Jordan <col_header> Venue </col_header> </cell> <cell>
Jordan <col_header> Opponent </col_header> </cell>
<cell> 5-0 <col_header> Result </col_header> </cell>
</table>
```

On 13 November 2013 Cristhian Stuani netted the second in a 5–0 win in Jordan.

Kale et al., INLG 2020

Visual Description

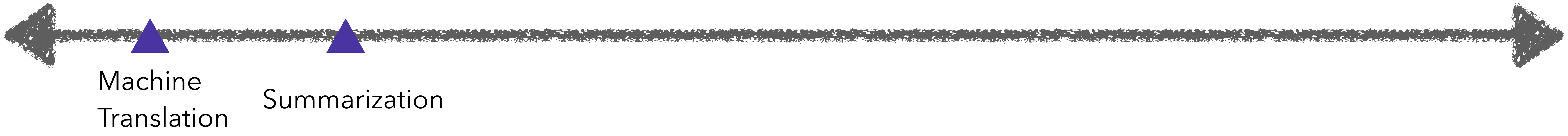


Two children are sitting at a table in a restaurant. The children are one little girl and one little boy. The little girl is eating a pink frosted donut with white icing lines on top of it. The girl has blonde hair and is wearing a green jacket with a black long sleeve shirt underneath. The little boy is wearing a black zip up jacket and is holding his finger to his lip but is not eating. A metal napkin dispenser is in between them at the table. The wall next to them is white brick. Two adults are on the other side of the short white brick wall. The room has white circular lights on the ceiling and a large window in the front of the restaurant. It is daylight outside.

Krause et al. CVPR 2017

Categorization of NLG tasks

Spectrum of open-endedness for NLG tasks



Machine Translation

Summarization

Source Sentence: 새해 복 많이 받으세요!

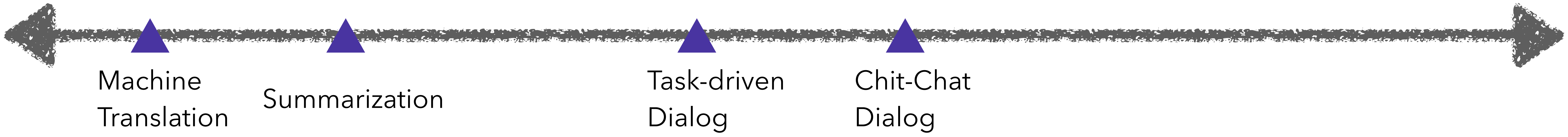
Reference Translations:

1. Happy new year!
2. Wish you a great year ahead!
3. Have a prosperous new year!

The output space is not diverse.

Categorization of NLG tasks

Spectrum of open-endedness for NLG tasks



Machine Translation

Summarization

Task-driven Dialog

Chit-Chat Dialog

Input: Hey, how are you doing?

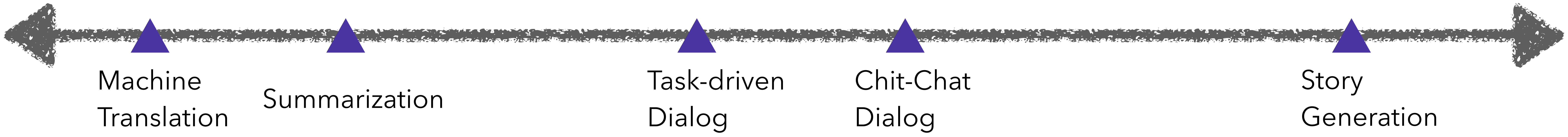
Reference Outputs:

1. Good, you?
2. I just heard an exciting news, do you want to hear it?
3. Thanks for asking! Barely surviving my homeworks.

The output space is getting more diverse...

Categorization of NLG tasks

Spectrum of open-endedness for NLG tasks



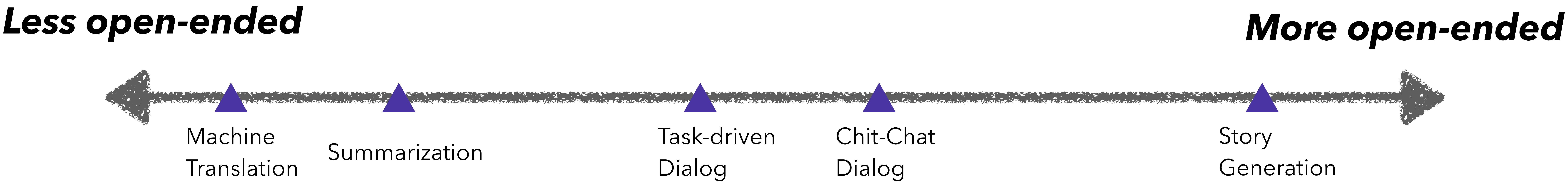
Input: Write a story about three little pigs?

Reference Outputs:

... (so may options)...

The output space is extremely diverse.

Categorization of NLG tasks



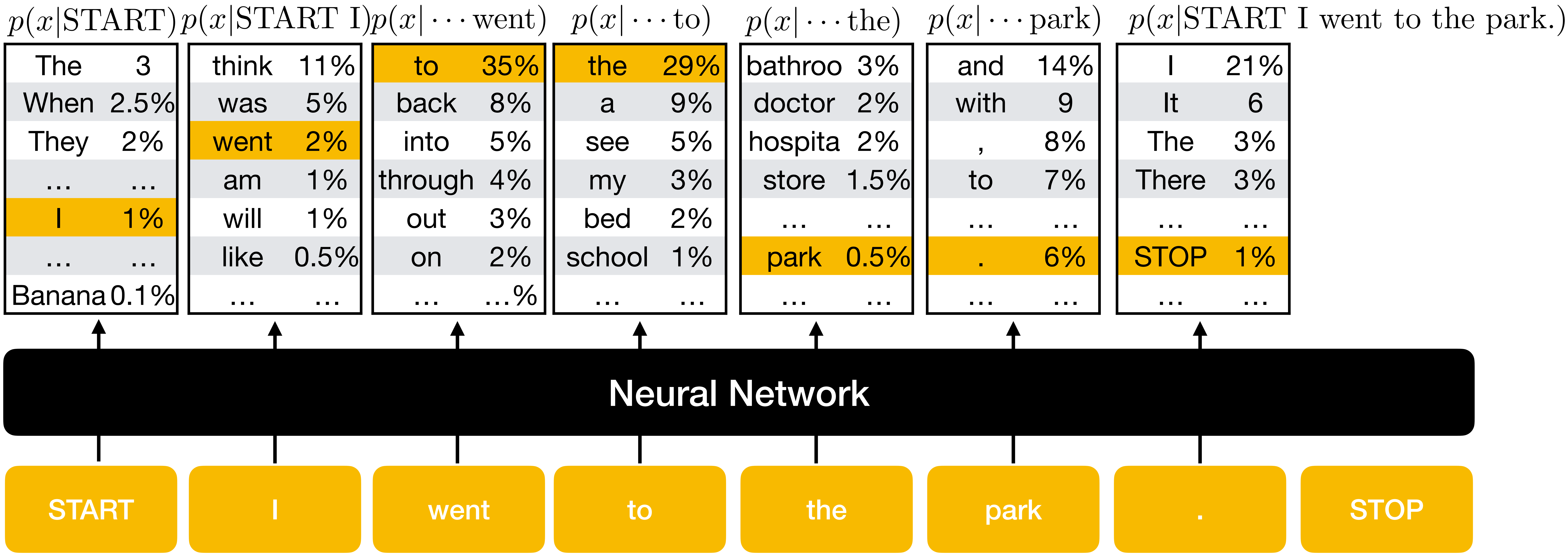
Less open-ended generation: the input mostly determines the correct output generation.

More open-ended generation: the output distribution still has high degree of freedom.

Remark: One way of formalizing categorization is *entropy*.
Tasks with different characteristics require different decoding and/or training approaches!

Neural language models

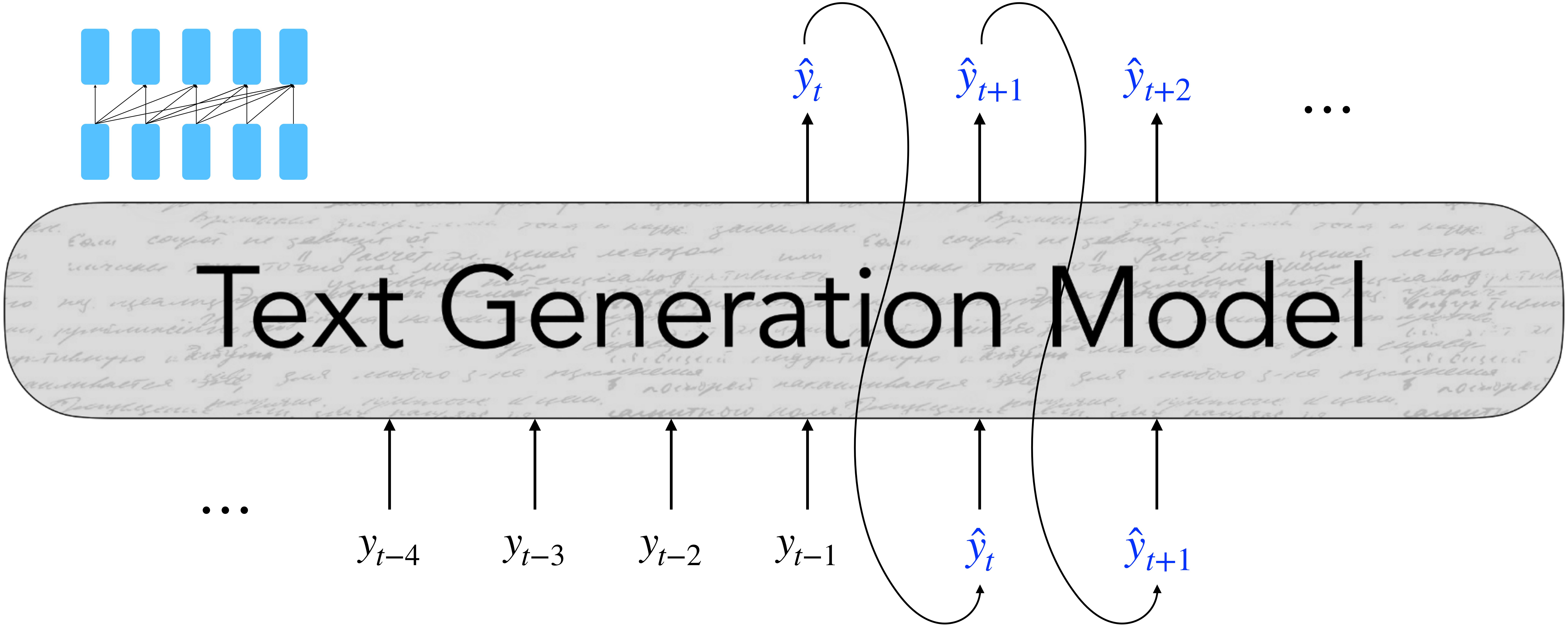
- **Input:** sequences of words (or tokens)
- **Output:** probability distribution over the next word (token)



$p(x\|\text{START})$		$p(x\|\text{START I})$		$p(x\|\cdots\text{went})$		$p(x\|\cdots\text{to})$		$p(x\|\cdots\text{the})$		$p(x\|\cdots\text{park})$		$p(x\|\text{START I went to the park.})$	
The	3	think	11%	to	35%	the	29%	bathroo	3%	and	14%	I	21%
When	2.5%	was	5%	back	8%	a	9%	doctor	2%	with	9	It	6
They	2%	went	2%	into	5%	see	5%	hospita	2%	,	8%	The	3%
…	…	am	1%	through	4%	my	3%	store	1.5%	to	7%	There	3%
I	1%	will	1%	out	3%	bed	2%	…	…	…	…	…	…
…	…	like	0.5%	on	2%	school	1%	park	0.5%	.	6%	STOP	1%
Banana	0.1%	…	…	…	…%	…	…	…	…	…	…	…	…

Autoregressive NLG with LLMs

- In autoregressive (decoder-only) LLMs, at each time step *t*, our model takes in a sequence of tokens as input $\{y\}_{<t}$ and outputs a new token, $\hat{y}_t$



Text Generation Model

Autoregressive NLG with LLMs

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

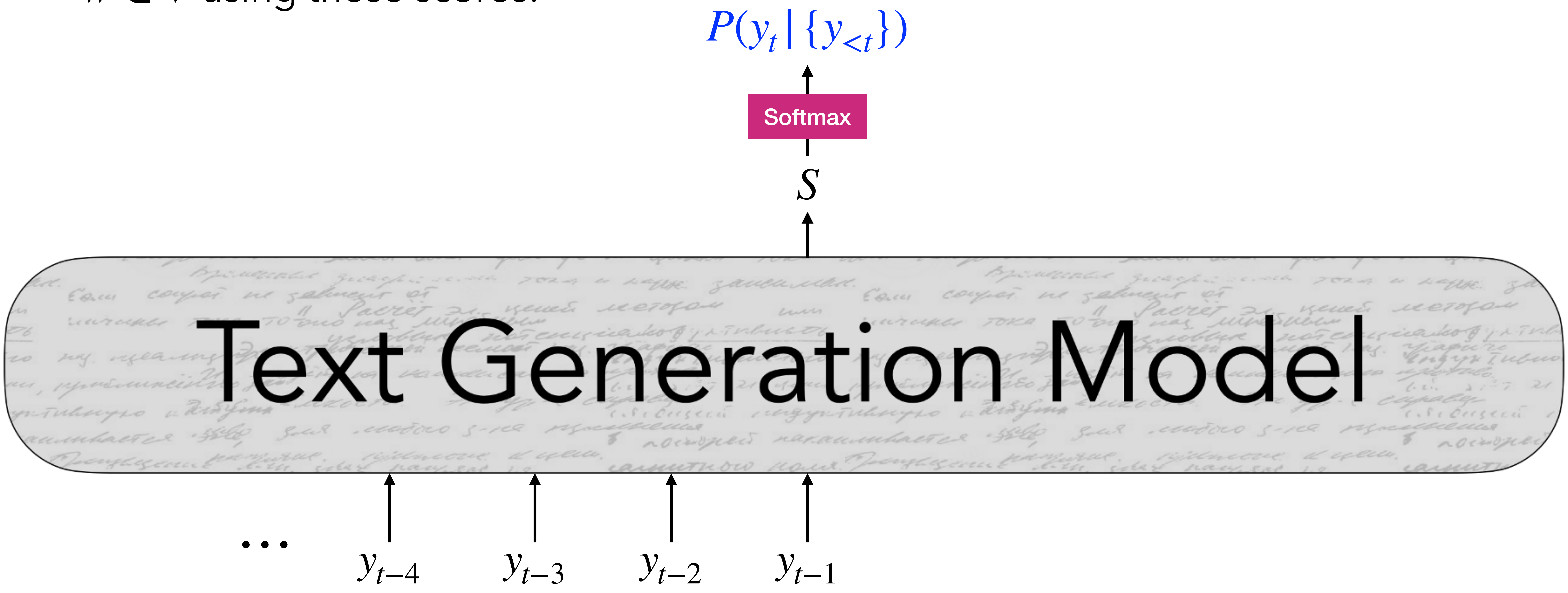
$$S = \underbrace{f(\{y_{<t}\}; \theta)}_{f(\,\cdot\,;\theta) \text{ is your model}}$$

- Then, we compute a probability distribution P over $w \in V$ using these scores:

$$P(y_t = w \mid \{y_{<t}\}) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

A look at a single step

- At each time step t, our model computes a vector of scores for each token in our vocabulary, $S \in \mathbb{R}^V$. Then, we compute a probability distribution P over $w \in V$ using these scores:



Recap: training and inference LLMs

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

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

$g(\cdot)$ is your decoding algorithm

  - An "obvious" decoding algorithm is to greedily choose the token with the highest probability at each time step

- At train time, we train the model to minimize the negative log-likelihood of the next token in the given sequence:

$$L_t = -\log P(y_t^* \mid \{y_{<t}^*\})$$

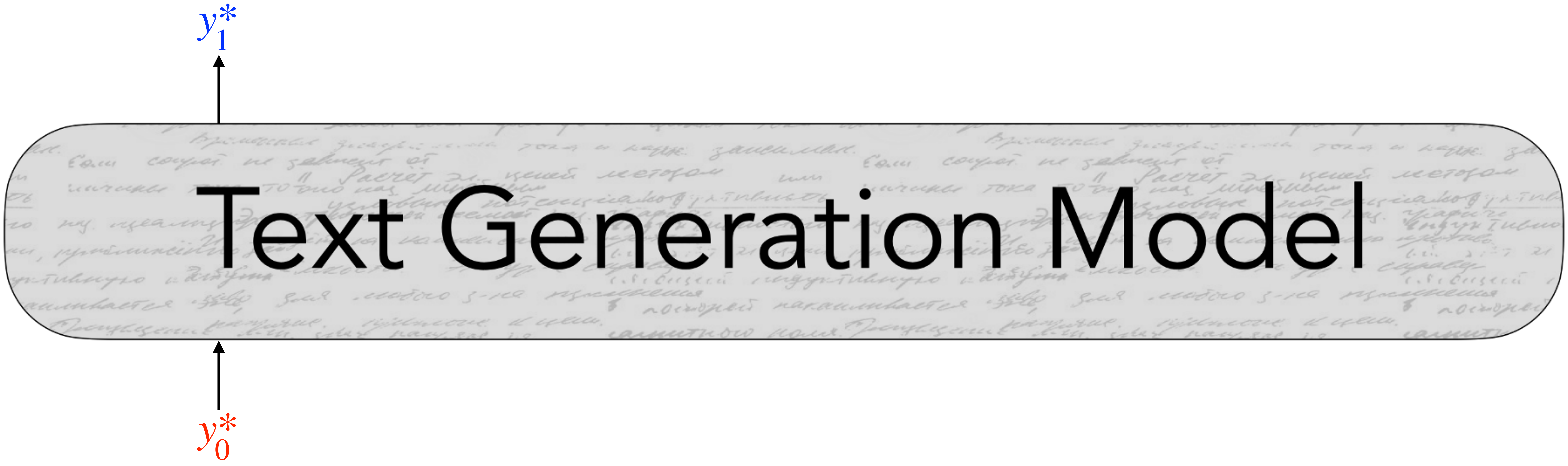
Remark:

- This is just a classification task where each $w \in V$ as a class.
- The label at each step is y_t^* in the training sequence.
- This token is often called "gold" or "ground-truth" token.
- This algorithm is often called "teacher-forcing".

Recap: Maximum Likelihood Training

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

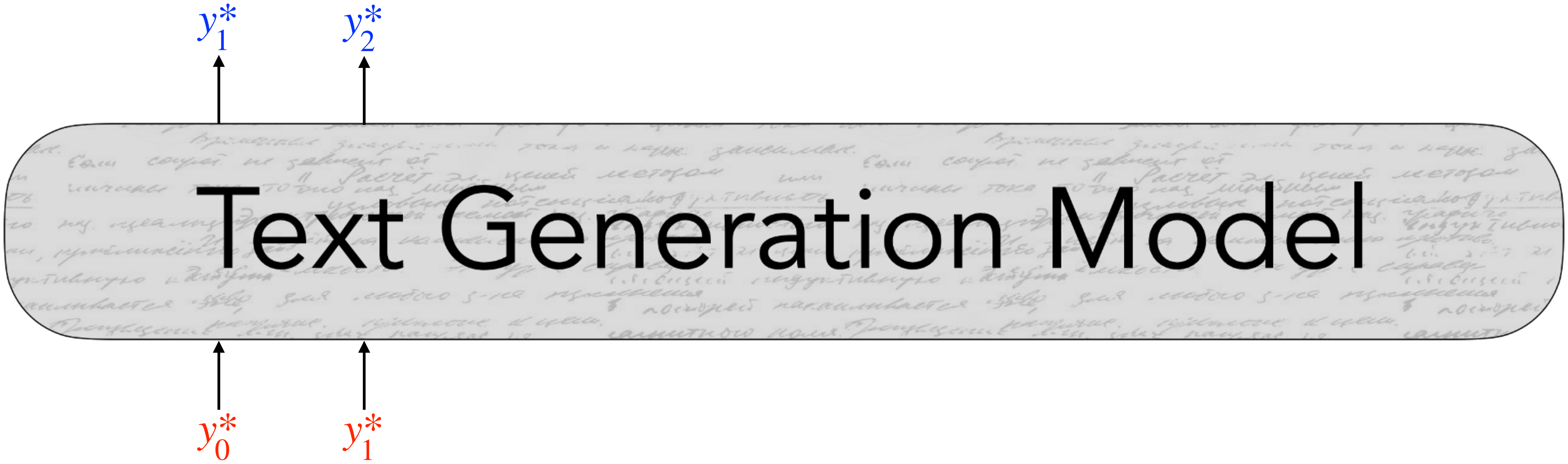
$$L = -\log P(y_1^* \mid y_0^*)$$



Recap: Maximum Likelihood Training

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

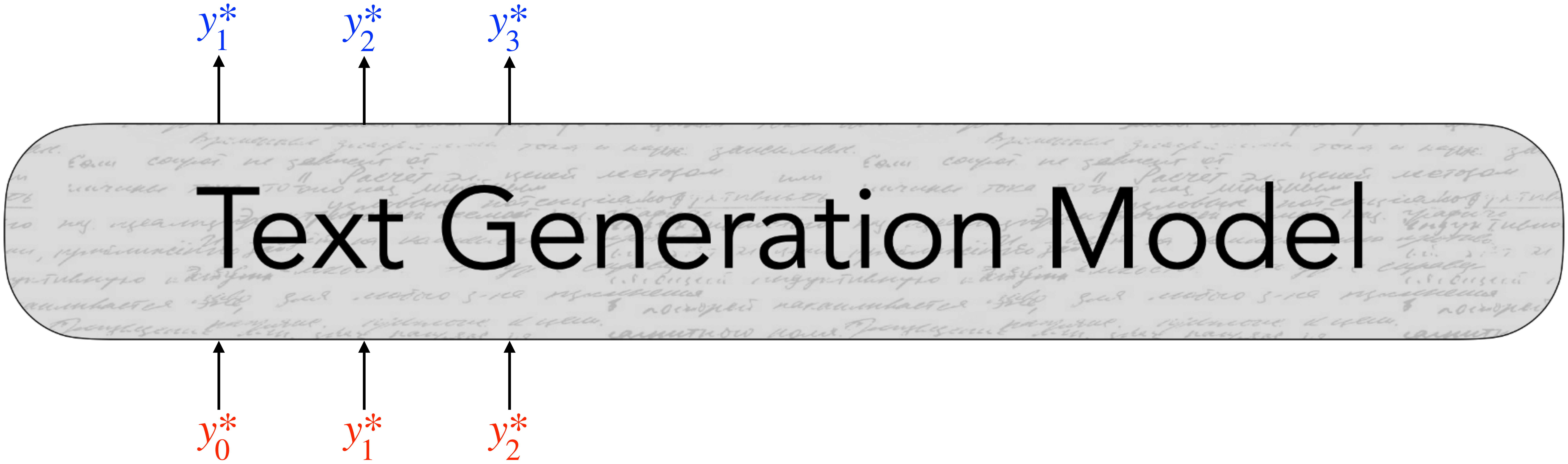
$$L = -\left(\log P(y_1^* \mid y_0^*) + \log P(y_2^* \mid y_0^*, y_1^*)\right)$$



Recap: Maximum Likelihood Training

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

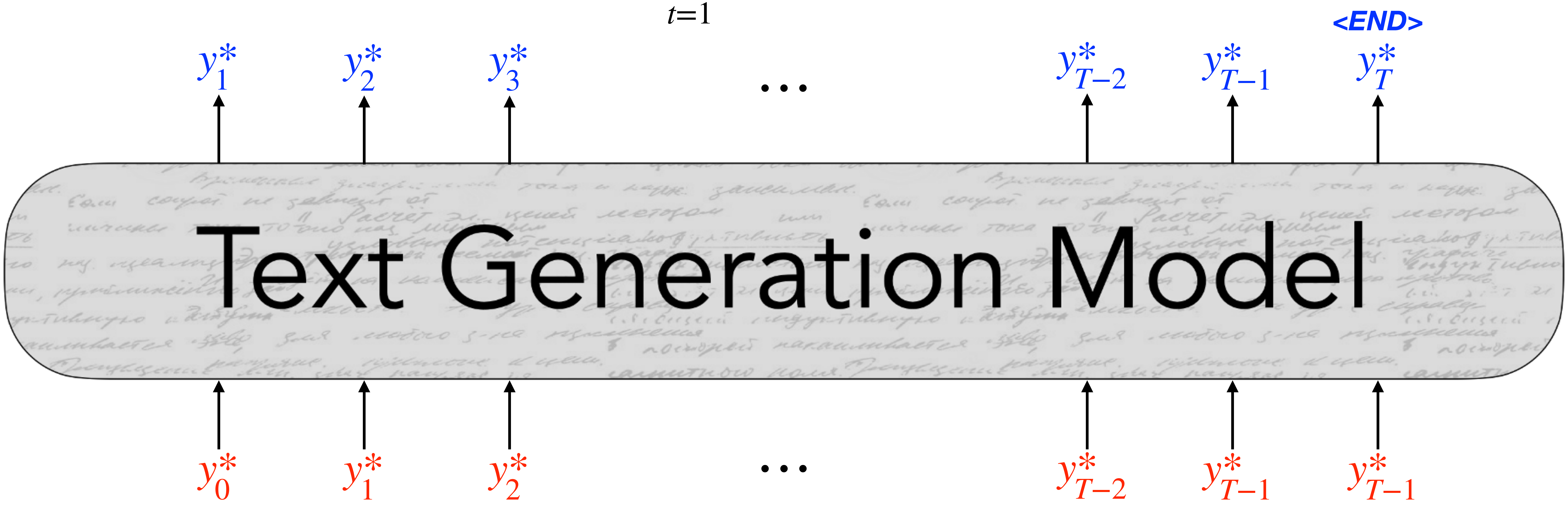
$$L = -\left(\log P(y_1^* \mid y_0^*) + \log P(y_2^* \mid y_0^*, y_1^*) + \log P(y_3^* \mid y_0^*, y_1^*, y_2^*)\right)$$



Recap: Maximum Likelihood Training

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

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



Decoding from LLMs

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

$$S = \underline{f(\{y_{<t}\};\theta)}$$

$f(\,\cdot\,;\theta)$ is your model

- Then, we compute a probability distribution P over $w \in V$ using these scores:

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

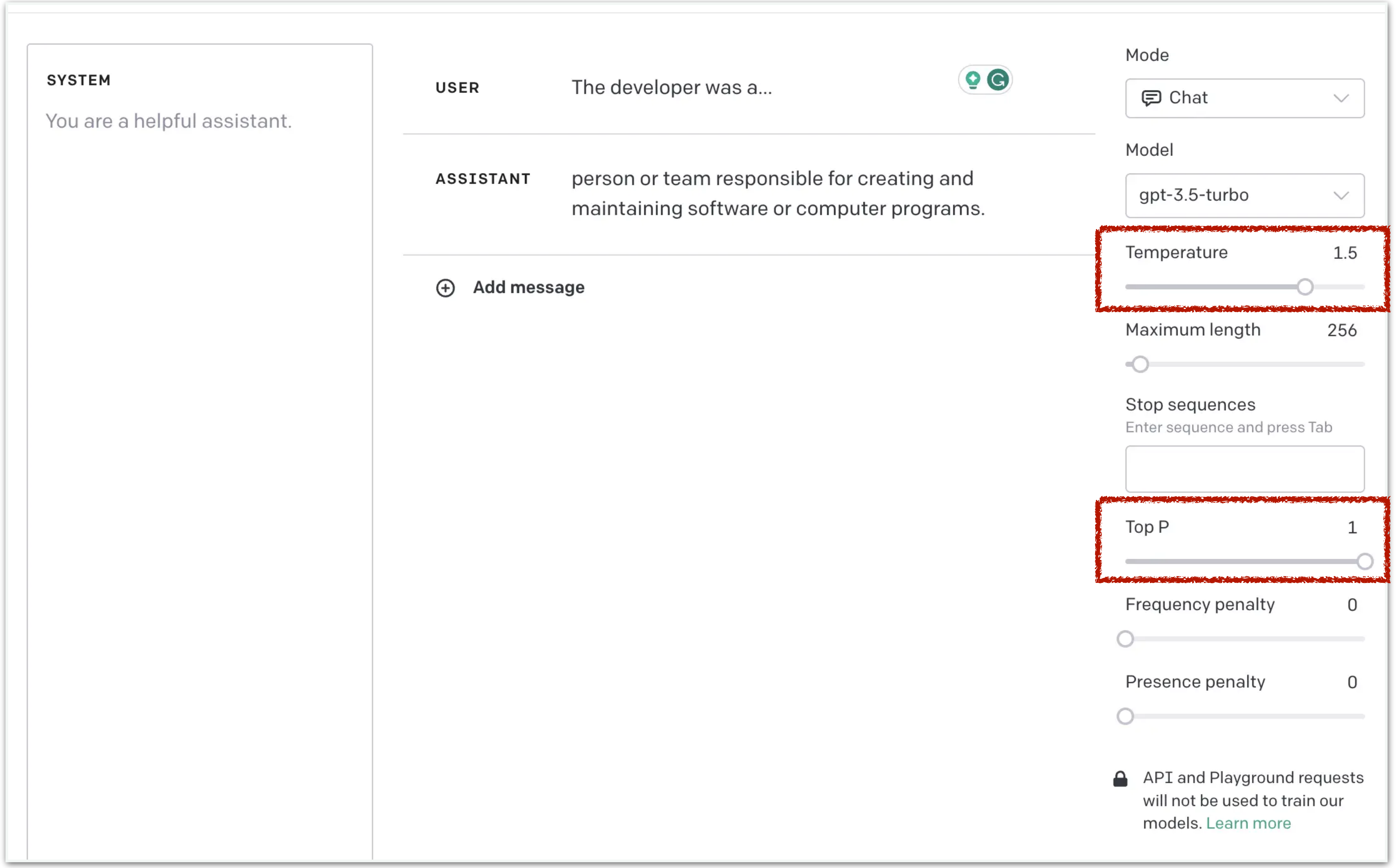
- Our decoding algorithm defines a function to select a token from this distribution:

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

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

Note: *we decode token by token from LLMs after they are trained (during inference)*

Decoding from ChatGPT



SYSTEM

You are a helpful assistant.

USER The developer was a...

ASSISTANT person or team responsible for creating and maintaining software or computer programs.

⊕ Add message

Mode: Chat

Model: gpt-3.5-turbo

Temperature 1.5

Maximum length 256

Stop sequences
Enter sequence and press Tab

Top P 1

Frequency penalty 0

Presence penalty 0

API and Playground requests will not be used to train our models. Learn more

ChatGPT API web interface

Note: *We will learn these decoding methods used in ChatGPT/GPT4 in this lecture!*

How to find the most likely text to generate?

- **Obvious method: Greedy Decoding**
  - Selects the highest probability token according to $P(y_t | y_{<t})$

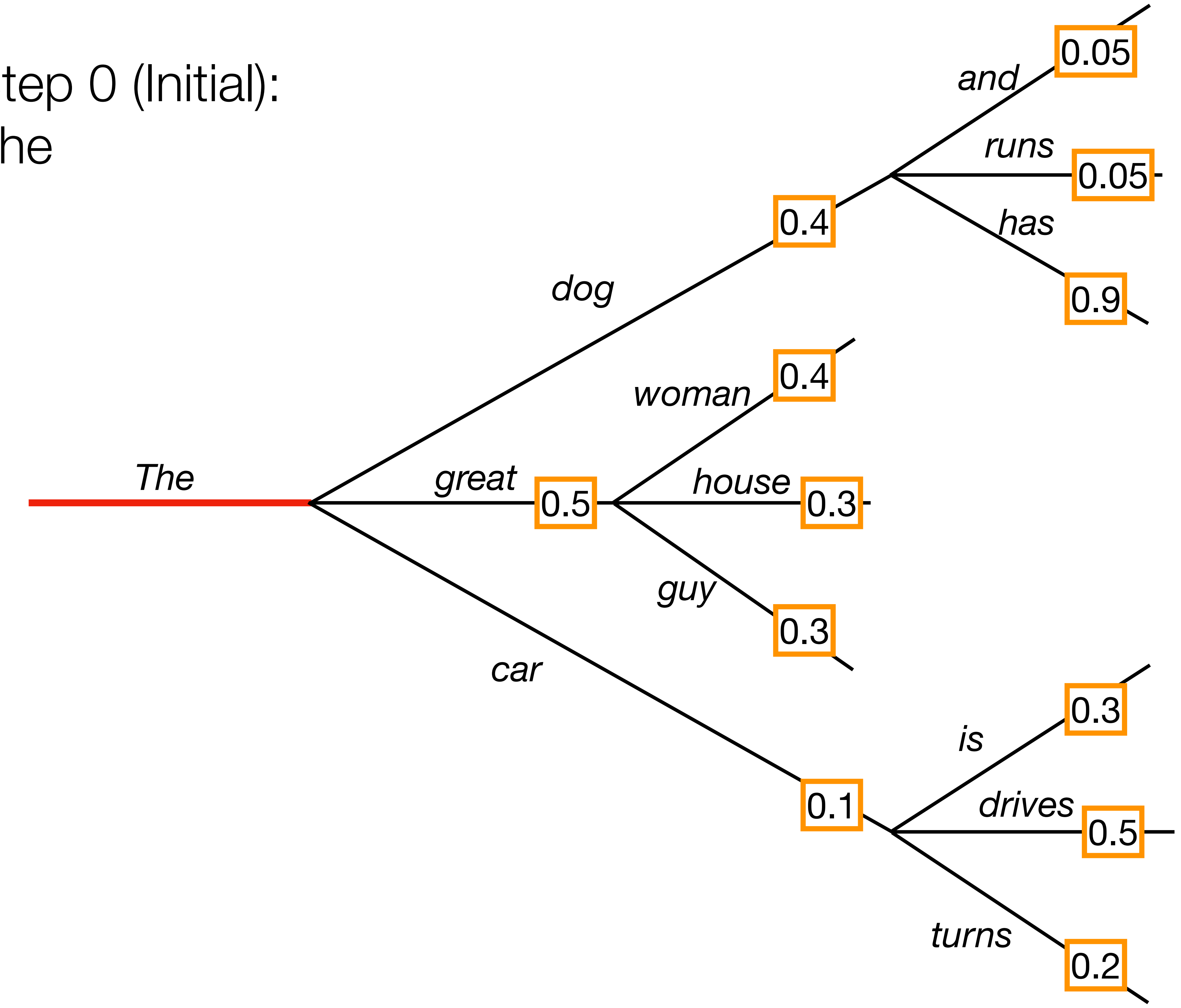
$$\hat{y}_t = \textbf{argmax}_{w \in V} P(y_t = w | y_{<t})$$

- **Beam Search**
  - Also aims to find the string with the highest probability, but with a wider exploration of candidates.

Greedy Decoding vs. Beam Search

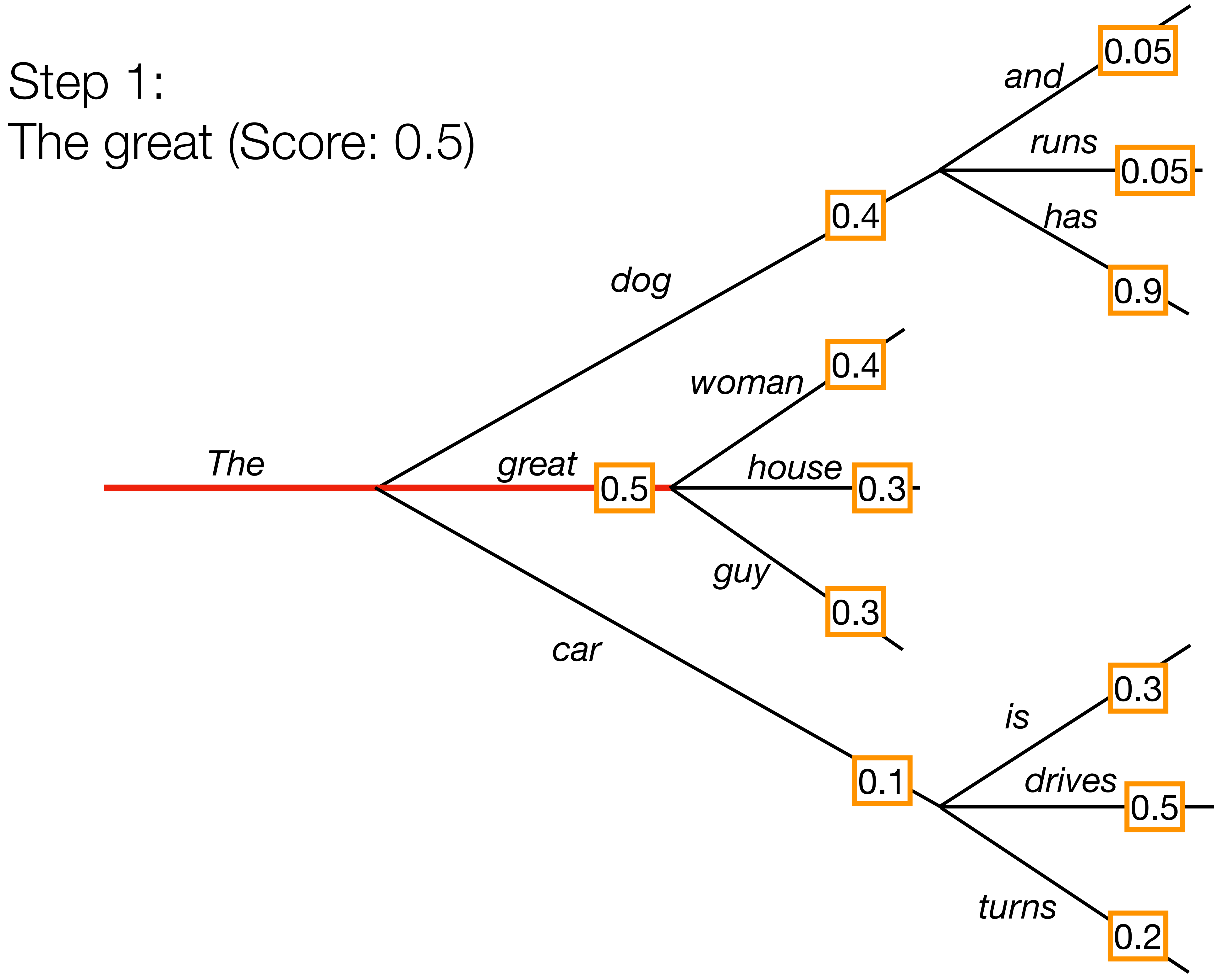
- **Greedy Decoding**
  - Choose the "currently best" token at each time step

Step 0 (Initial):
The



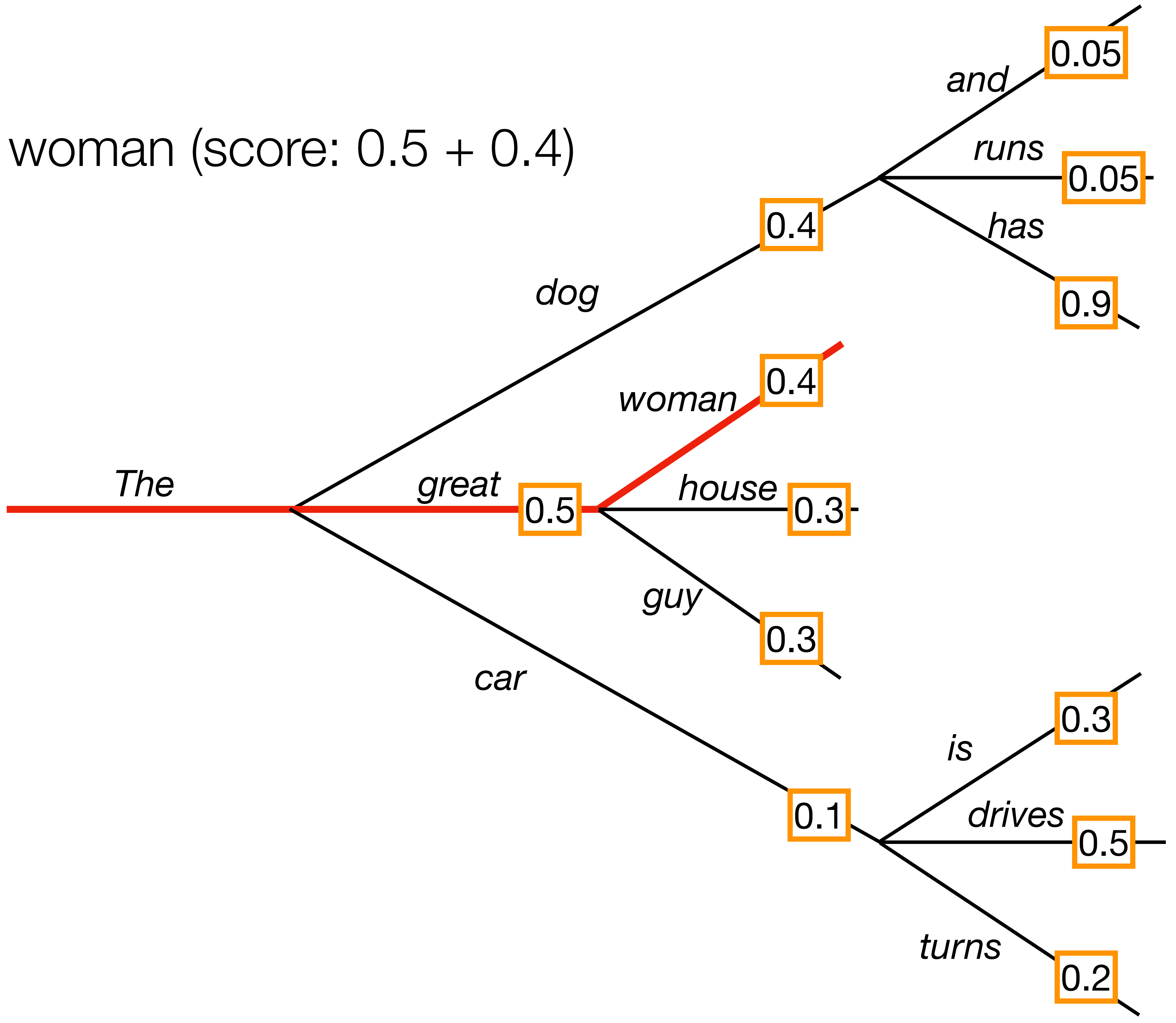
Greedy Decoding vs. Beam Search

- **Greedy Decoding**
  - Choose the "currently best" token at each time step



Greedy Decoding vs. Beam Search

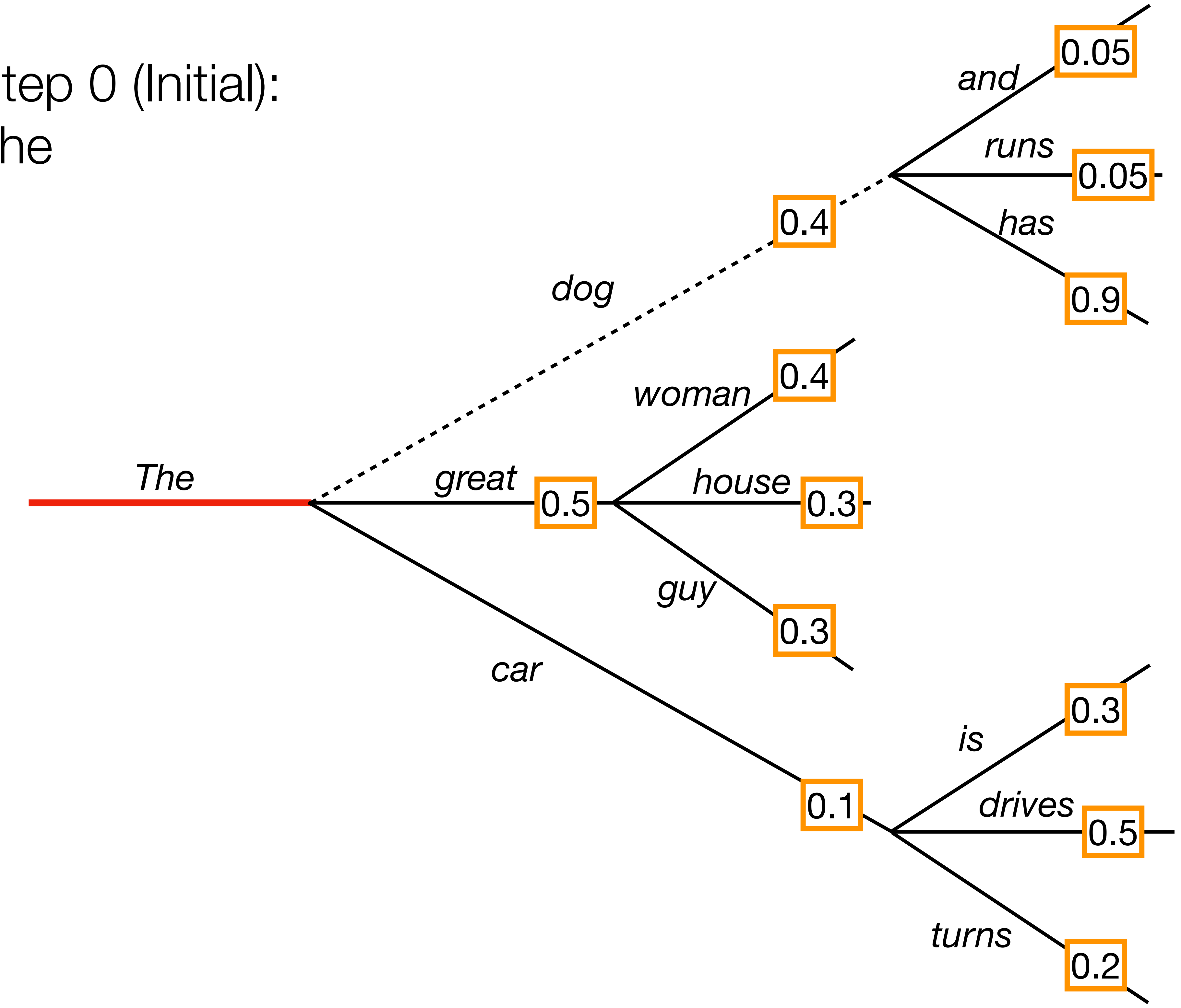
- **Greedy Decoding**
  - Choose the "currently best" token at each time step


Step 2:
The great woman (score: 0.5 + 0.4)

Greedy Decoding vs. Beam Search

- **Beam Search (in this example, *beam_width* = 2)**
  - At each step, retain 2 hypotheses with the highest probability

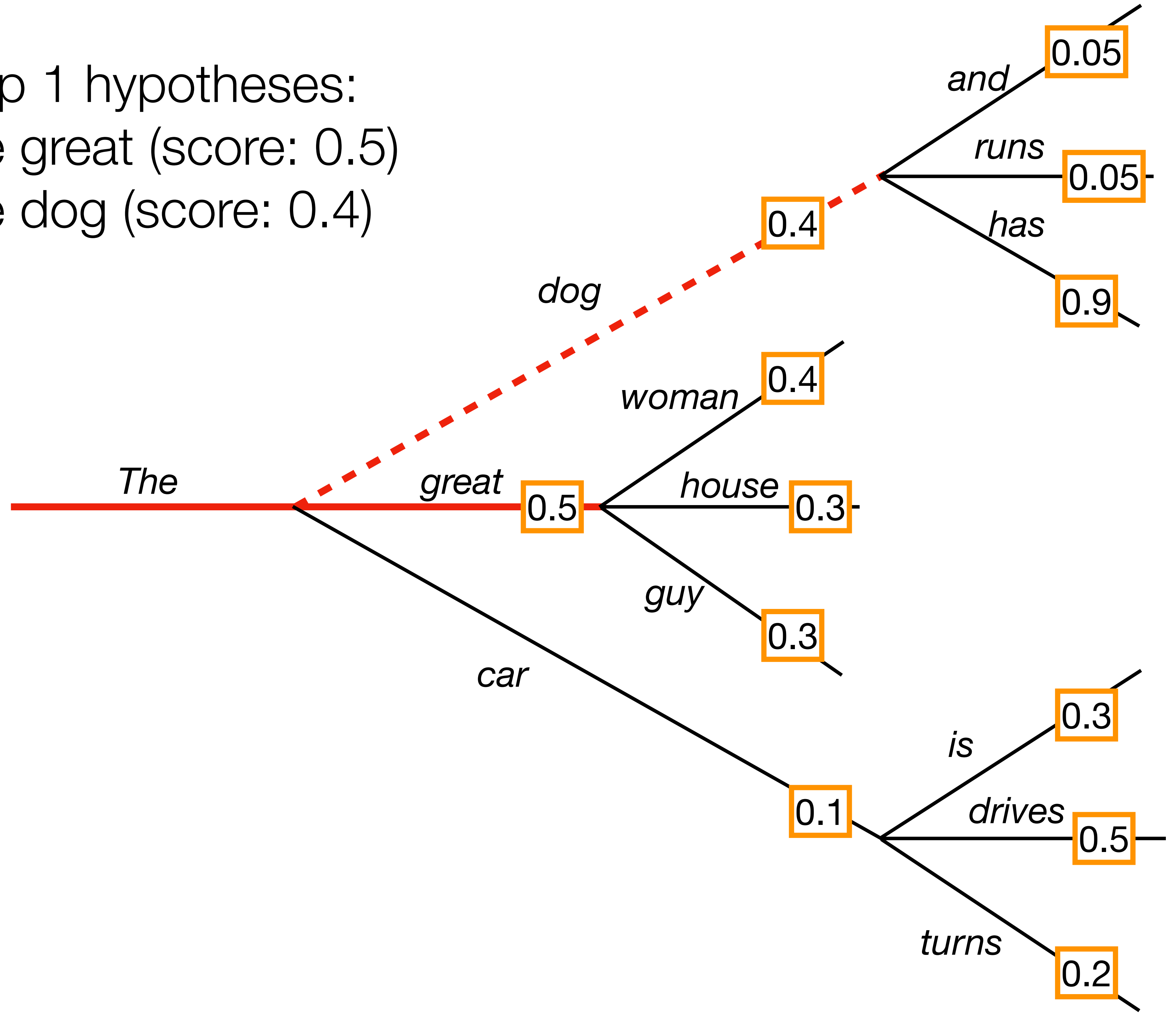
Step 0 (Initial):
The



Greedy Decoding vs. Beam Search

- **Beam Search (in this example, *beam_width* = 2)**
  - At each step, retain 2 hypotheses with the highest probability

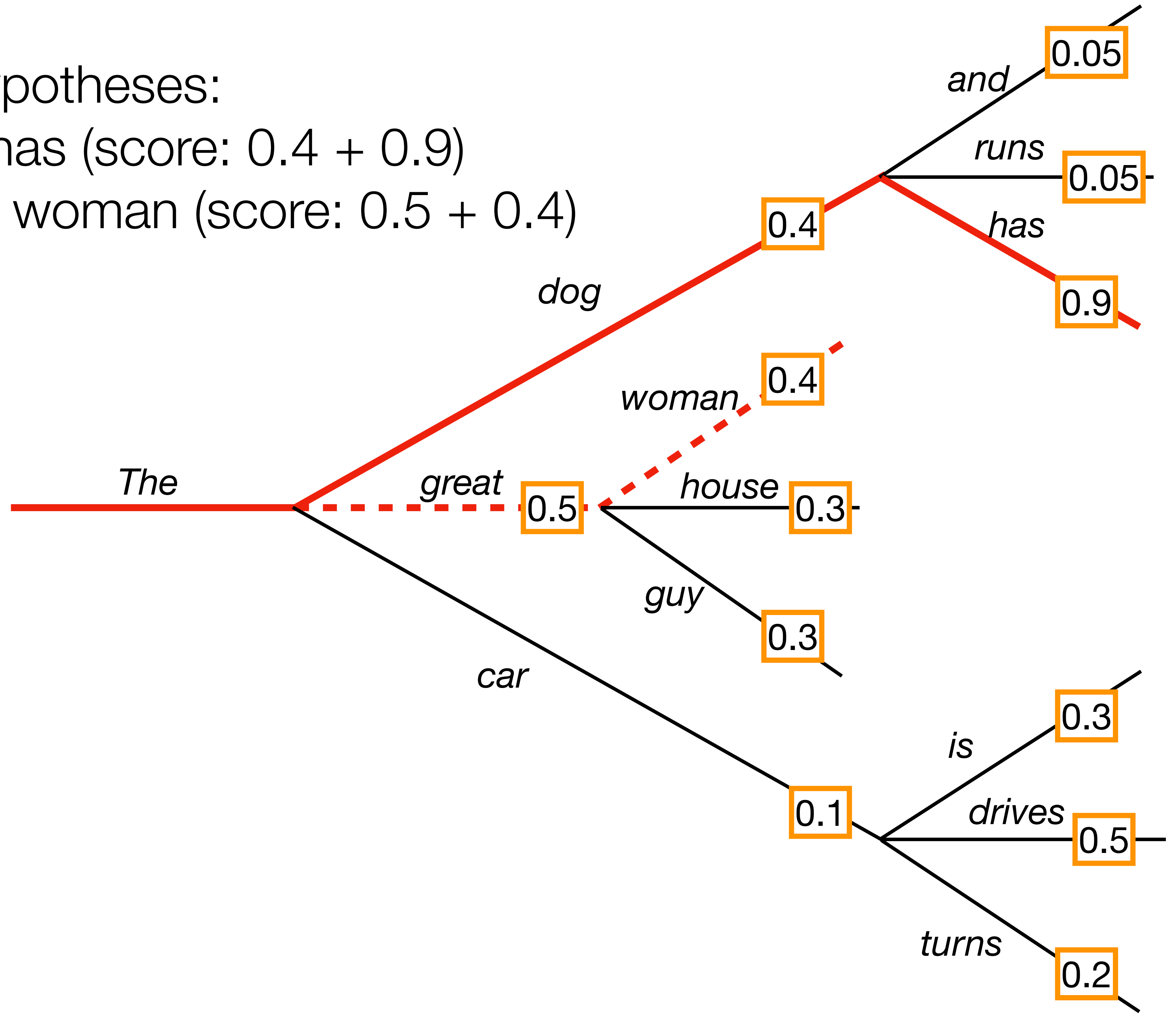
Step 1 hypotheses:
The great (score: 0.5)
The dog (score: 0.4)



Greedy Decoding vs. Beam Search

- **Beam Search (in this example, *beam_width* = 2)**
  - At each step, retain 2 hypotheses with the highest probability

Step 2 hypotheses:
The dog has (score: 0.4 + 0.9)
The great woman (score: 0.5 + 0.4)



How to find the most likely text to generate?

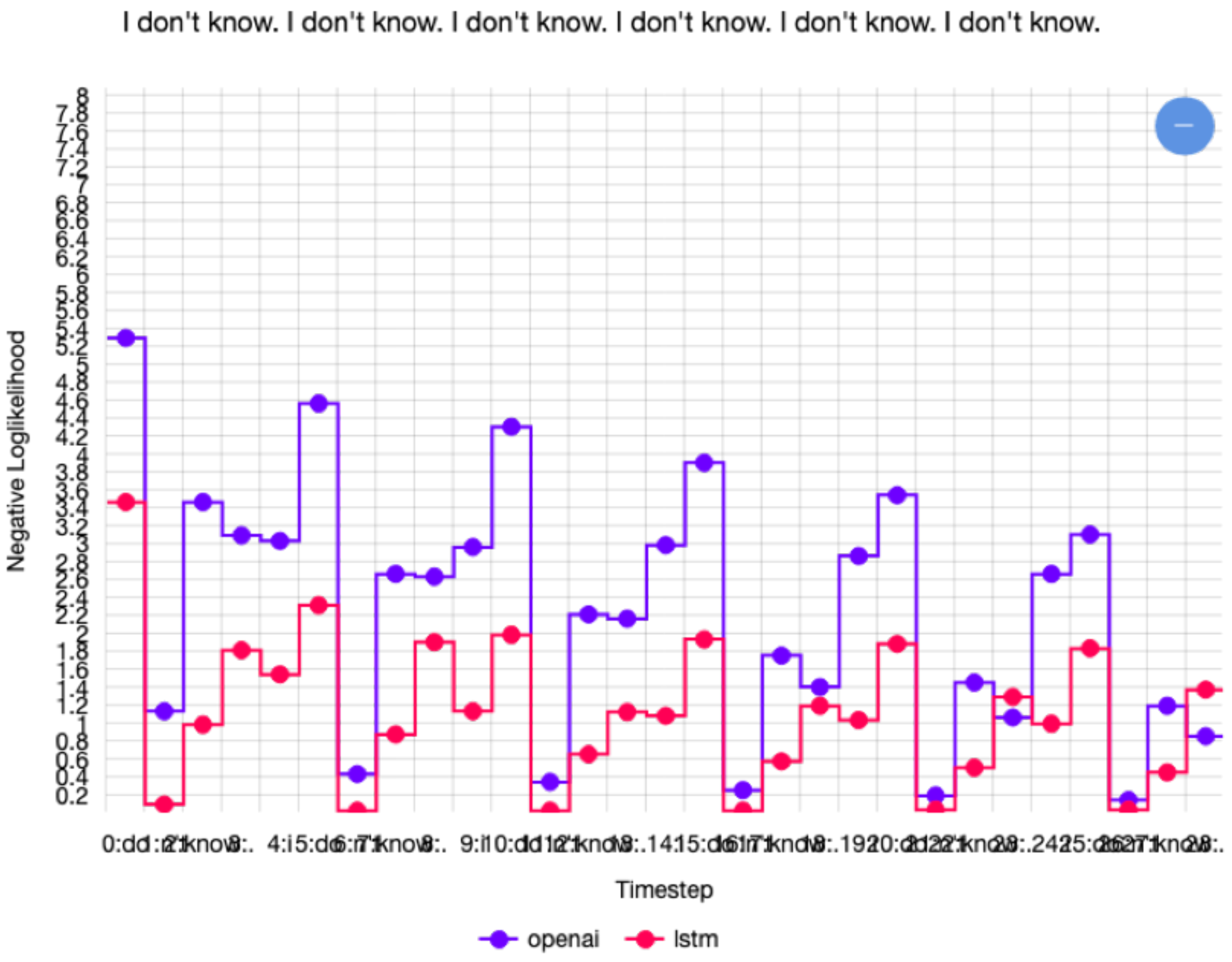
- **Beam Search**
  - A form of best-first-search for the most likely string, but with a wider exploration of candidates.
  - Compared to greedy decoding, beam search gives a better approximation of brute-force search over all sequences
  - A small overhead in computation due to beam width
    Time complexity: O(beam width * vocab size * generation length)

    * *Naive brute-force search: O(vocab size ^ generation length), hence intractable!*

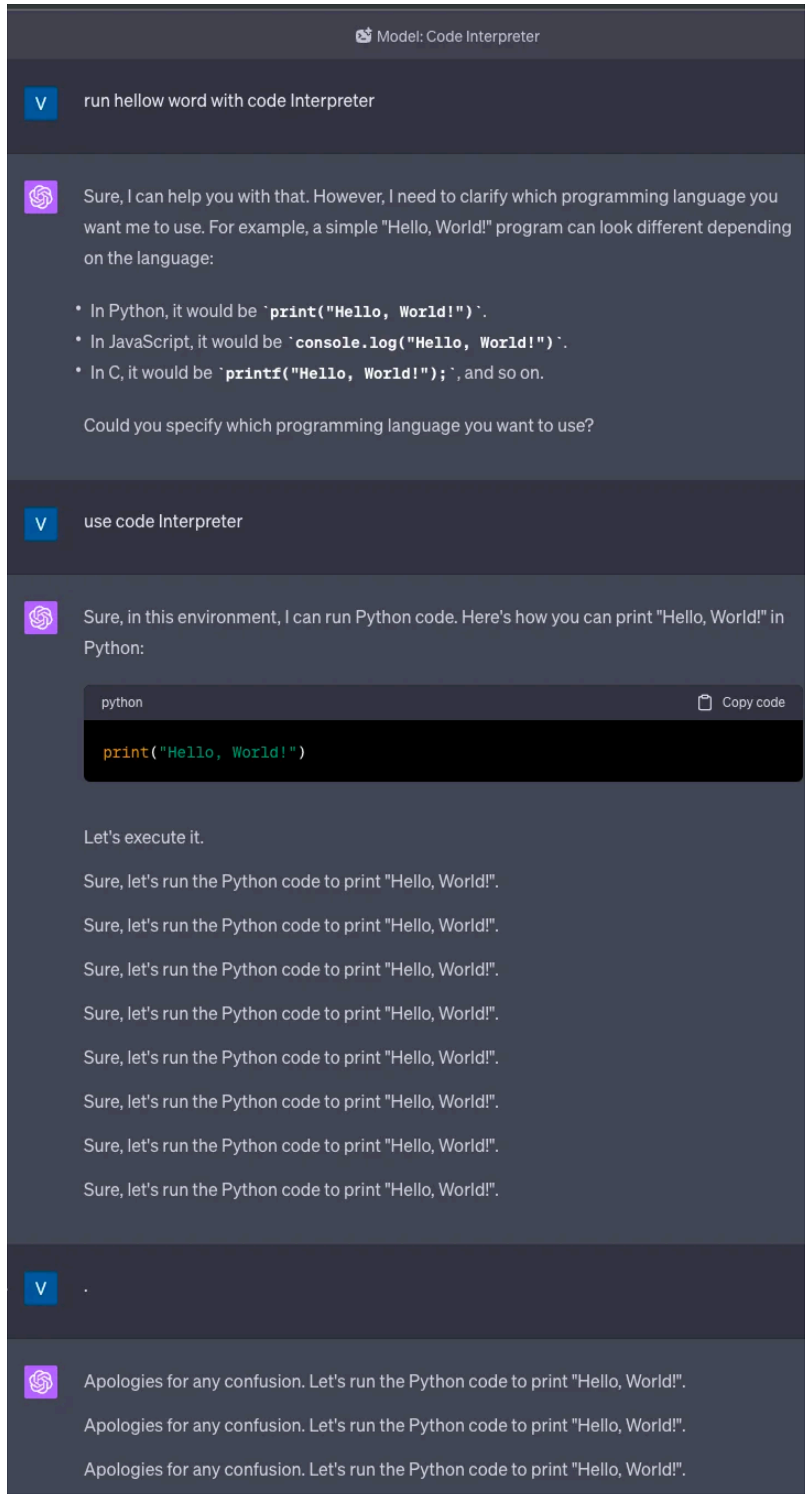
Note: *Overall, greedy / beam search is widely used for low-entropy tasks like MT and summarization.*

But, are greedy sequences always the best solution? 🤔

But, most likely sequences are repetitive…



Probability of "I don't know" increases with each repetition, creating a positive feedback loop.



Model: Code Interpreter

V run hellow word with code Interpreter

Sure, I can help you with that. However, I need to clarify which programming language you want me to use. For example, a simple "Hello, World!" program can look different depending on the language:

- In Python, it would be `print("Hello, World!")`.
- In JavaScript, it would be `console.log("Hello, World!")`.
- In C, it would be `printf("Hello, World!");`, and so on.

Could you specify which programming language you want to use?

V use code Interpreter

Sure, in this environment, I can run Python code. Here's how you can print "Hello, World!" in Python:

```python
print("Hello, World!")
```

Let's execute it.

Sure, let's run the Python code to print "Hello, World!".

Sure, let's run the Python code to print "Hello, World!".

Sure, let's run the Python code to print "Hello, World!".

Sure, let's run the Python code to print "Hello, World!".

Sure, let's run the Python code to print "Hello, World!".

Sure, let's run the Python code to print "Hello, World!".

Sure, let's run the Python code to print "Hello, World!".

Sure, let's run the Python code to print "Hello, World!".

V .

Apologies for any confusion. Let's run the Python code to print "Hello, World!".

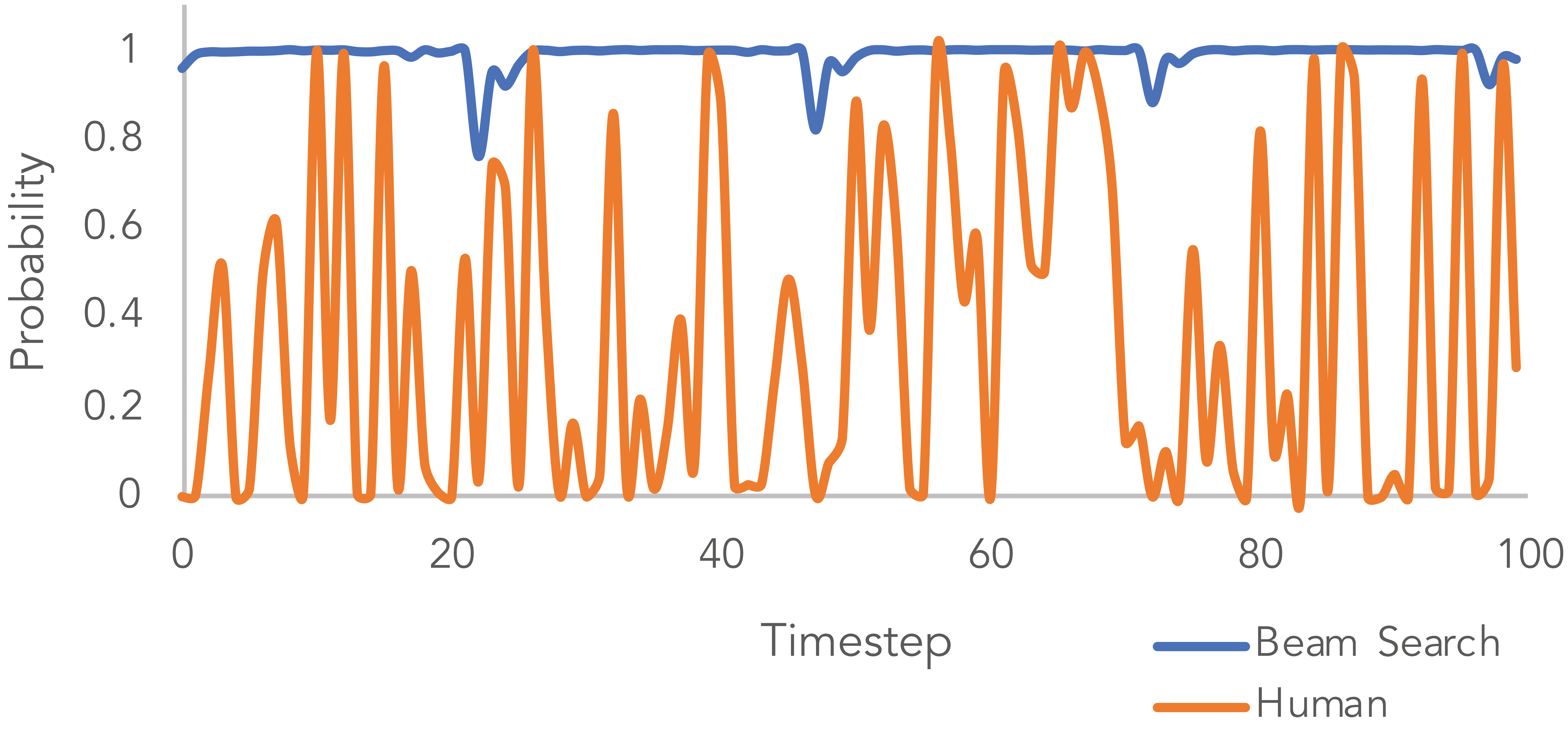
Apologies for any confusion. Let's run the Python code to print "Hello, World!".

Apologies for any confusion. Let's run the Python code to print "Hello, World!".

https://chat.openai.com/share/4d8eb91f-fe1c-430e-bdd3-cafd434ec3d4

(Holtzman et al. ICLR 2020)

Also, are greedy methods reasonable for open-ended generation?



Greedy methods fail to capture the variance of human text distribution.

(Holtzman et al. ICLR 2020)

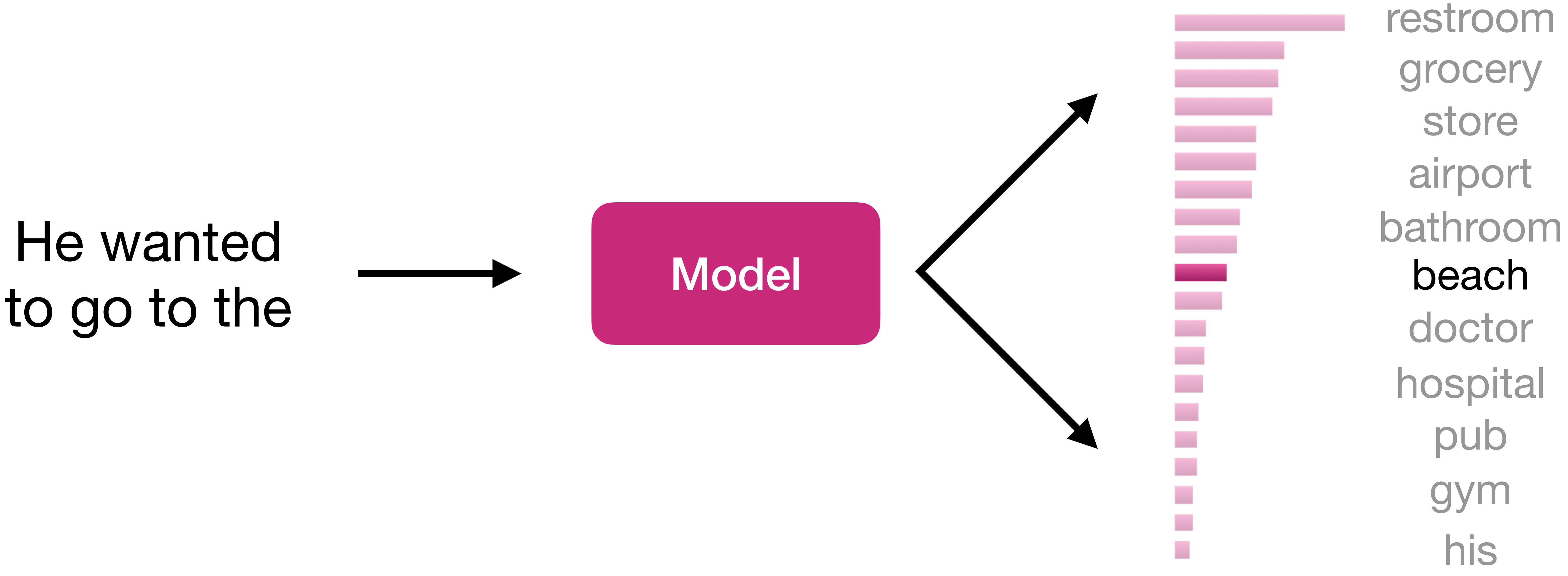
Sampling generation from LLMs

Time to get random: Sampling

- Sample a token from the token distribution at each step!

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

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



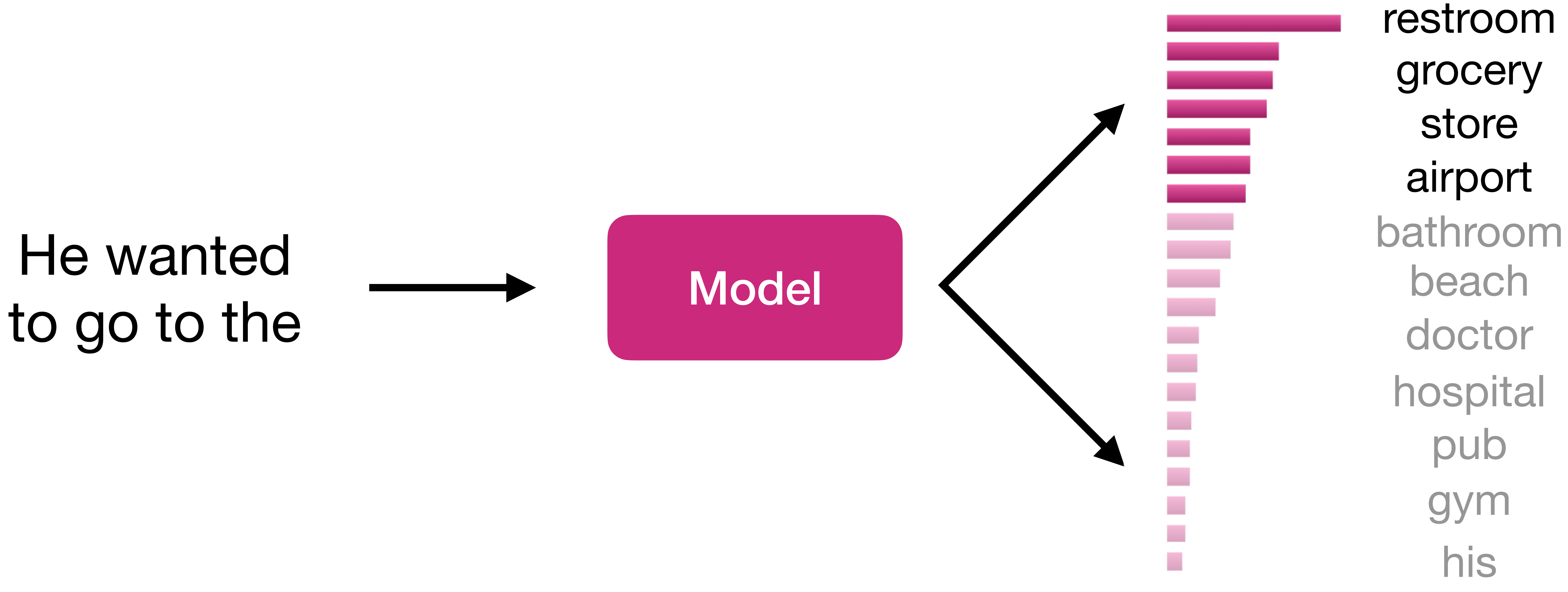
Decoding: Top-k Sampling

- 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 and in aggregate have considerable mass (statistics speak: we have "heavy tailed" distributions)
  - Many tokens are probably really wrong in the current context.
  - Although *each of them* may be assigned a small probability, *in aggregate* they still get a high chance to be selected.

- Solution: Top-*k* sampling *(Fan et al., 2018)*
  - Only sample from the top *k* tokens in the probability distribution.

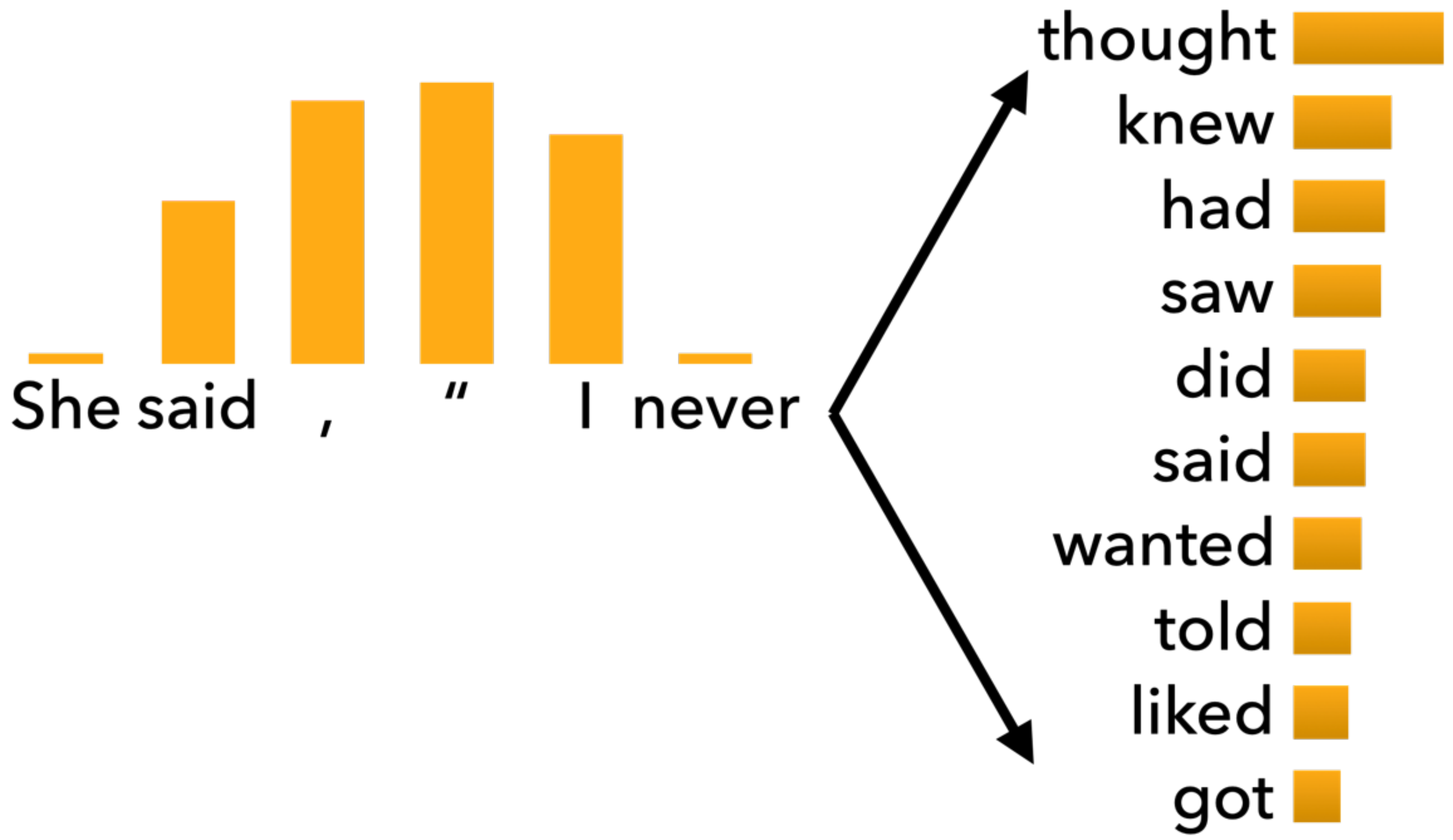
Decoding: Top-k Sampling

- Solution: Top-*k* sampling *(Fan et al., 2018)*
  - Only sample from the top *k* tokens in the probability distribution.
  - Common values for *k* = 10, 20, 50 (*but it's up to you!*)



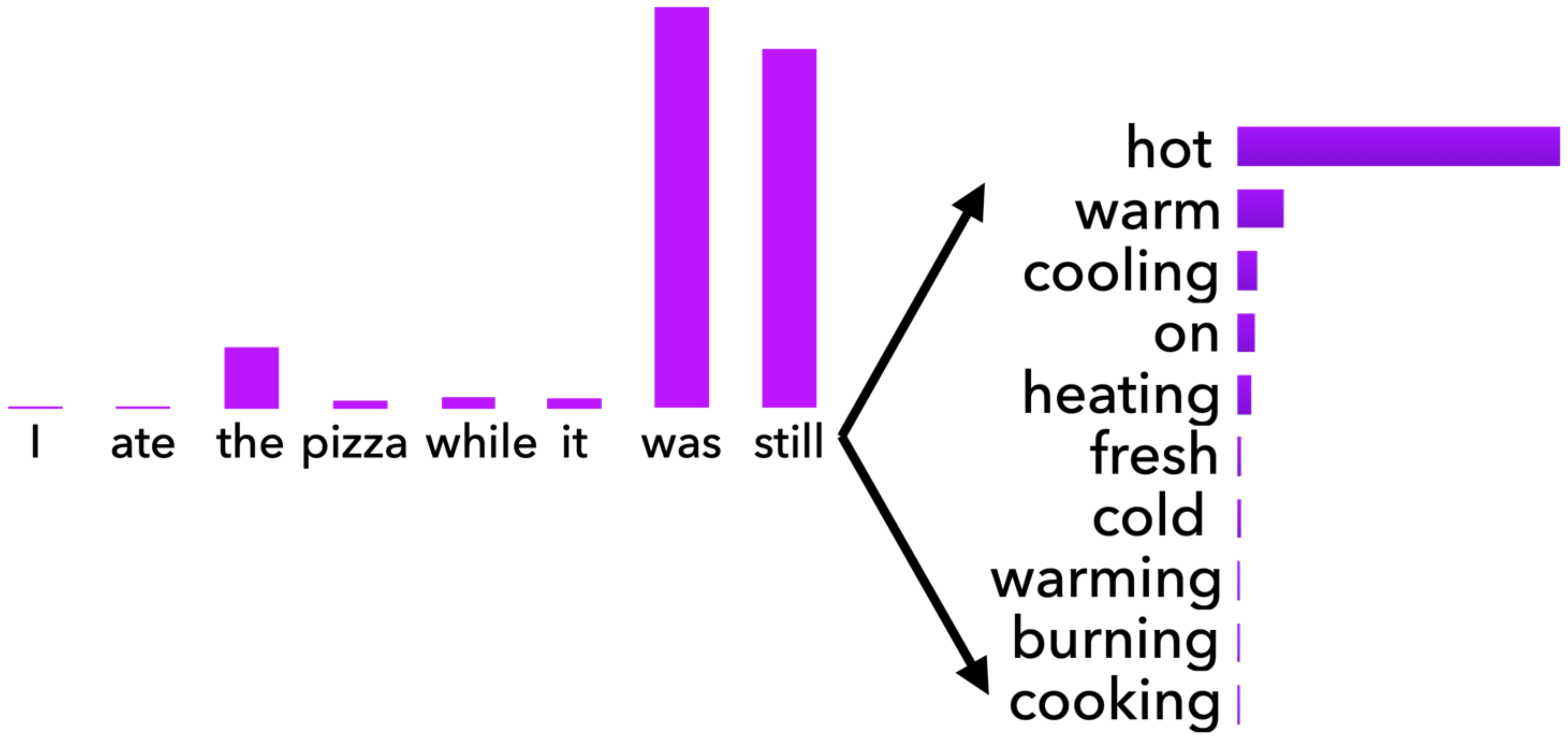
  - Increasing *k* yields more **diverse**, but **risky** outputs
  - Decreasing *k* yields more **safe** but **generic** outputs

Issues with Top-k Sampling



For *flat* distribution,

Top-*k* Sampling may cut off too **quickly**!



For *peaked* distribution,

Top-*k* Sampling may also cut off too **slowly**!

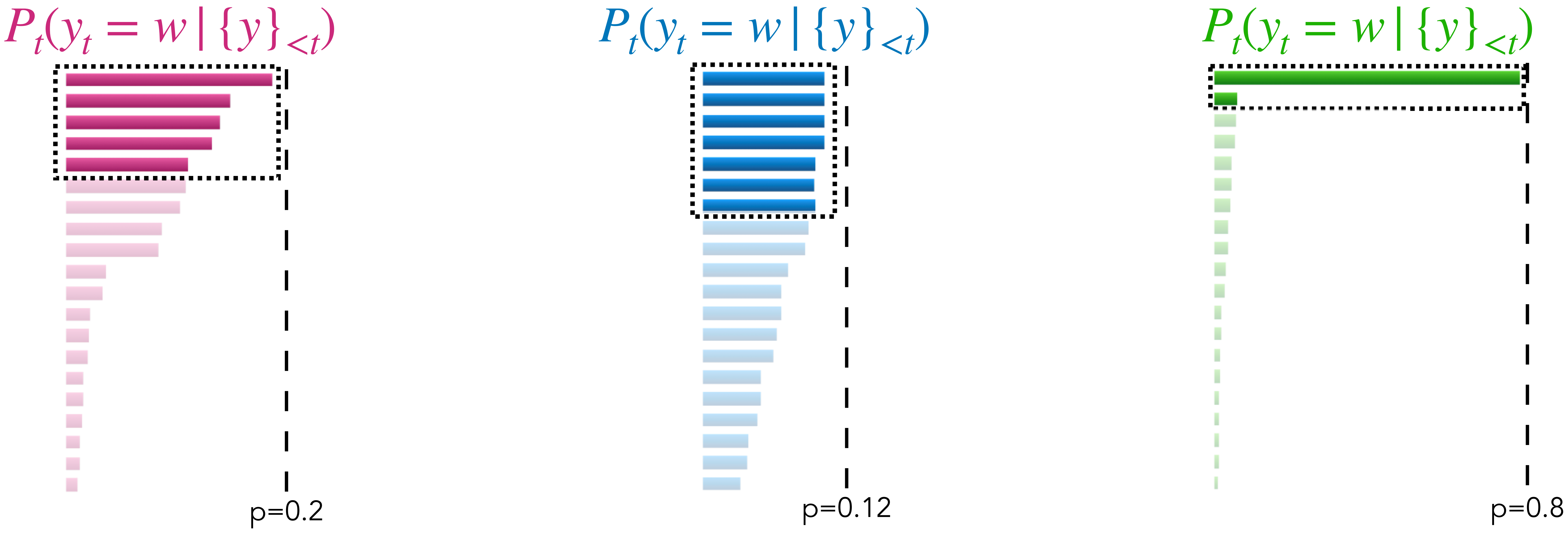
Decoding: Top-p (Nucleus) Sampling

- Problem: The token distributions we sample from are dynamic
  - When the distribution P_t is flat, small k removes many viable options.
  - When the distribution P_t is peaked, large k allows too many options a chance to be selected.

- Solution: Top-*p* sampling *(Holtzman et al., 2020)*
  - Sample from all tokens in the top p cumulative probability mass (i.e., where mass is concentrated)
  - Varies k according to the uniformity of P_t

Decoding: Top-p (Nucleus) Sampling

- Solution: Top-*p* sampling *(Holtzman et al., 2020)*
  - Sample from all tokens in the top p cumulative probability mass (i.e., where mass is concentrated)
  - Varies k according to the uniformity of P_t



Scaling randomness: Softmax temperature

- Recall: At time step t, model computes a distribution P_t by applying softmax to a vector of scores $S \in \mathbb{R}^{|V|}$

$$P_t(y_t = w \mid \{y_{<t}\}) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- Here, you can apply **temperature hyperparameter** τ to the softmax to rebalance P_t:

$$P_t(y_t = w \mid \{y_{<t}\}) = \frac{\exp(S_w/\tau)}{\sum_{w' \in V} \exp(S_{w'}/\tau)}$$

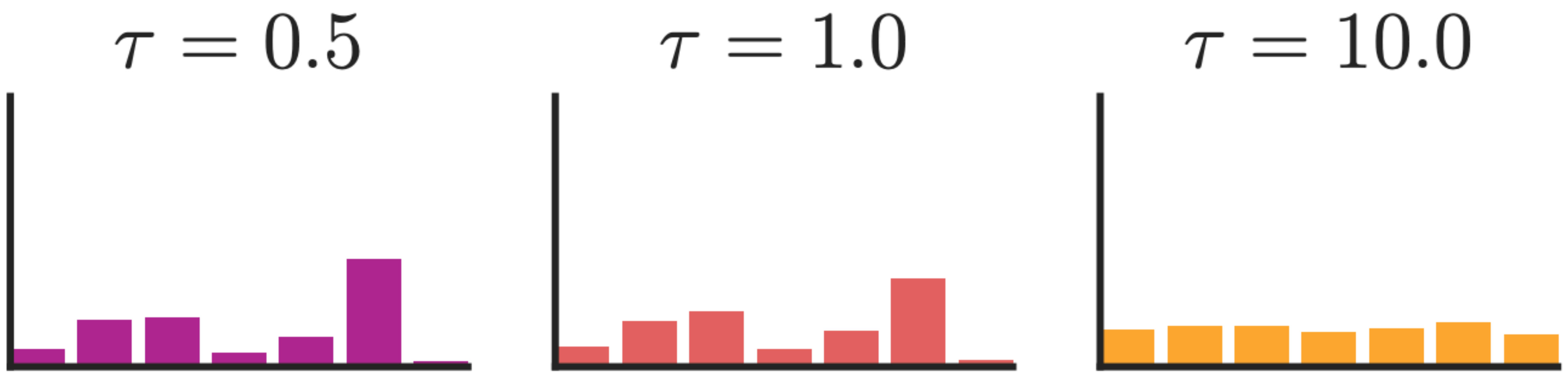
- Raise the temperature $\tau > 1$: P_t becomes more uniform
  - More diverse output (probability is spread across vocabulary)
- Lower the temperature $\tau < 1$: P_t becomes more spiky
  - Less diverse output (probability concentrated to the top tokens)

Scaling randomness: Softmax temperature

- You can apply **temperature hyperparameter** τ to the softmax to rebalance P_t:

$$P_t(y_t = w \mid \{y_{<t}\}) = \frac{\exp(S_w/\tau)}{\sum_{w' \in V} \exp(S_{w'}/\tau)}$$

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



Scaling randomness: Softmax temperature

- You can apply **temperature hyperparameter** τ to the softmax to rebalance P_t:

$$P_t(y_t = w \mid \{y_{<t}\}) = \frac{\exp(S_w/\tau)}{\sum_{w' \in V} \exp(S_{w'}/\tau)}$$

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

NOTE: Temperature is a hyperparameter for decoding algorithm, not an algorithm itself! It can be applied for both beam search and sampling methods.

Toward better generation: Re-ranking

- Problem: What if I already have decoded a bad sequence from my model?

- **Decode a bunch of sequences**
  - Sample $n = 10, 20, 50, \ldots$ sequences with the same input given
- Define a score to approximate quality of sequences and **re-rank by this score**
  - Simplest score: (low) perplexity
    - Careful! Remember that even the repetitive sequences get low perplexity in general…
  - Re-rankers can evaluate a variety of properties:
    - Style *(Holtzman et al., 2018)*, Discourse *(Gabriel et al., 2021)*, Factuality *(Goyal et al., 2020)*, Logical Consistency *(Jung et al. 2022)*, and many more
  - Can compose multiple re-rankers together.

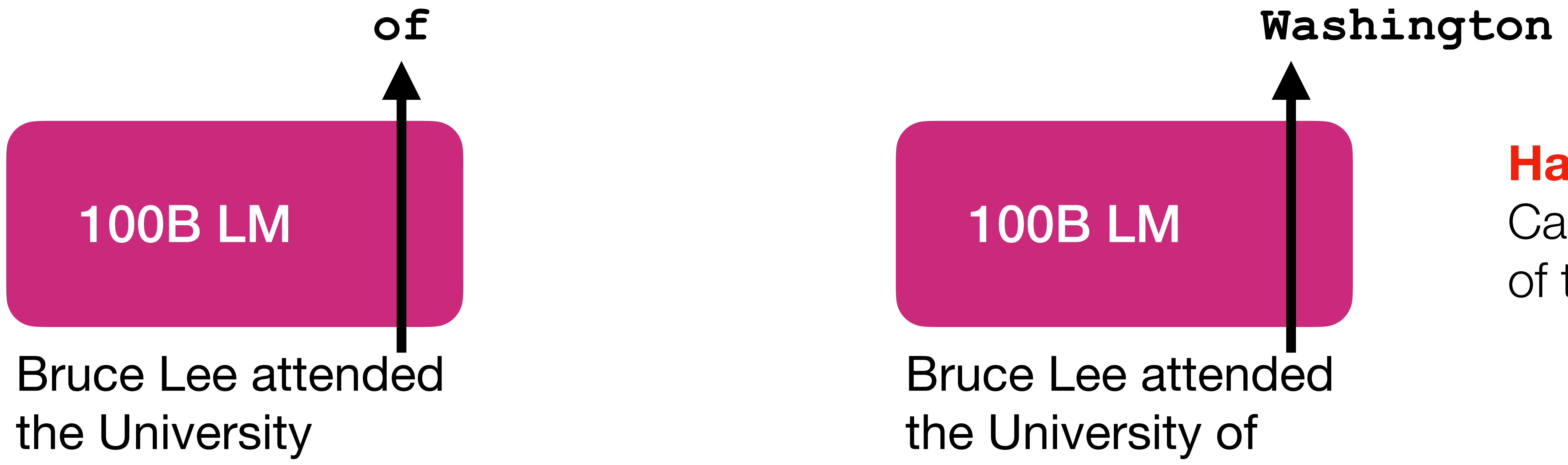
Speeding-up generation from LLMs

Speeding-up generation: Speculative Sampling

- Problem: Generating with a large LM takes a long time

- Intuition: Not all tokens are equally hard to generate!

Easy to predict:
May be a 1B LM
can predict this too


of
100B LM
Bruce Lee attended the University
Washington
100B LM
Bruce Lee attended the University of

Hard to predict:
Can really make use
of the 100B LM here

- **Idea**: Use a generation from small LM to assist large LM generation

** Same idea independently proposed from DeepMind and Google - see Chen et al., 2023; Leviathan et al., 2023*

Speeding-up generation: Speculative Sampling

- First, sample a draft of length K (= 5 in this example) from a small LM M_p

$$y_1 \sim p(\cdot \mid \underbrace{x}_{\text{Input prefix}}), y_2 \sim p(\cdot \mid x, y_1), \cdots, y_5 \sim p(\cdot \mid x, y_1, y_2, y_3, y_4)$$

- Then, compute the token distribution at each time step with a large target LM M_q

$$q(\cdot \mid x), q(\cdot \mid x, y_1), \underbrace{q(\cdot \mid x, y_1, y_2)}_{\text{Next token distribution of } M_q \text{, when given } x, y_1, y_2}, \cdots, q(\cdot \mid x, y_1, \cdots, y_5)$$

    - Note: This can be computed in a *single forward pass* of M_q (Why?)

- Let's denote $p_i = p(\cdot \mid x, y_1, \cdots, y_{i-1})$ and $q_i = q(\cdot \mid x, y_1, \cdots y_{i-1})$
  e.g., $q_2 = q(\cdot \mid x, y_1)$, *i.e. next token distribution predicted by the target model* M_q, *when given* x *and* y_1

Speeding-up generation: Speculative Sampling

- Now, we can compare the probability of each token assigned by draft model M_p and target model M_q

	Token	y_1	y_2	y_3	y_4	y_5
		dogs	love	chasing	after	cars
Draft model (1B)	p_i	0.8	0.7	0.9	0.8	0.7
Target model (100B)	q_i	0.9	0.8	0.8	0.3	0.8

- Starting from y_1, decide whether or not to accept the tokens generated by the draft model.

Speeding-up generation: Speculative Sampling

- Now, we can compare the probability of each token assigned by draft model M_p and target model M_q

	Token	y_1	y_2	y_3	y_4	y_5
		dogs	love	chasing	after	cars
Draft model (1B)	p_i	0.8	0.7	0.9	0.8	0.7
Target model (100B)	q_i	0.9	0.8	0.8	0.3	0.8

- Starting from y_1, decide whether or not to accept the tokens generated by the draft model.
- Case 1: $q_i \geq p_i$

  The target model (100B) likes this token, even more than the draft model (which generated it).

  => Accept this token!

Generation after step 1:
`dogs`

Speeding-up generation: Speculative Sampling

- Now, we can compare the probability of each token assigned by draft model M_p and target model M_q

	Token	y_1	y_2	y_3	y_4	y_5
		dogs	love	chasing	after	cars
Draft model (1B)	p_i	0.8	$\underline{0.7}$	0.9	0.8	0.7
Target model (100B)	q_i	0.9	$\underline{0.8}$	0.8	0.3	0.8

- Starting from y_1, decide whether or not to accept the tokens generated by the draft model.
- Case 1: $q_i \geq p_i$
  The target model (100B) likes this token, even more than the draft model (which generated it).
  => Accept this token!

Generation after step 2:
`dogs love`

Speeding-up generation: Speculative Sampling

- Now, we can compare the probability of each token assigned by draft model M_p and target model M_q

	Token	y_1	y_2	y_3	y_4	y_5
		dogs	love	chasing	after	cars
Draft model (1B)	p_i	0.8	0.7	0.9	0.8	0.7
Target model (100B)	q_i	0.9	0.8	0.8	0.3	0.8

- Case 2: $q_i < p_i$ (accept)

Target model doesn't like this token as much as the draft model...

=> Accept it with the probability $\dfrac{q_i}{p_i}$

Generation after step 3:

`dogs love chasing`

In this example, assume we accepted it with prob=0.8/0.9

Speeding-up generation: Speculative Sampling

- Now, we can compare the probability of each token assigned by draft model M_p and target model M_q

	Token	y_1	y_2	y_3	y_4	y_5
		dogs	love	chasing	aft	ars
Draft model (1B)	p_i	0.8	0.7	0.9	0.8	0.7
Target model (100B)	q_i	0.9	0.8	0.8	[illegible]	0.8

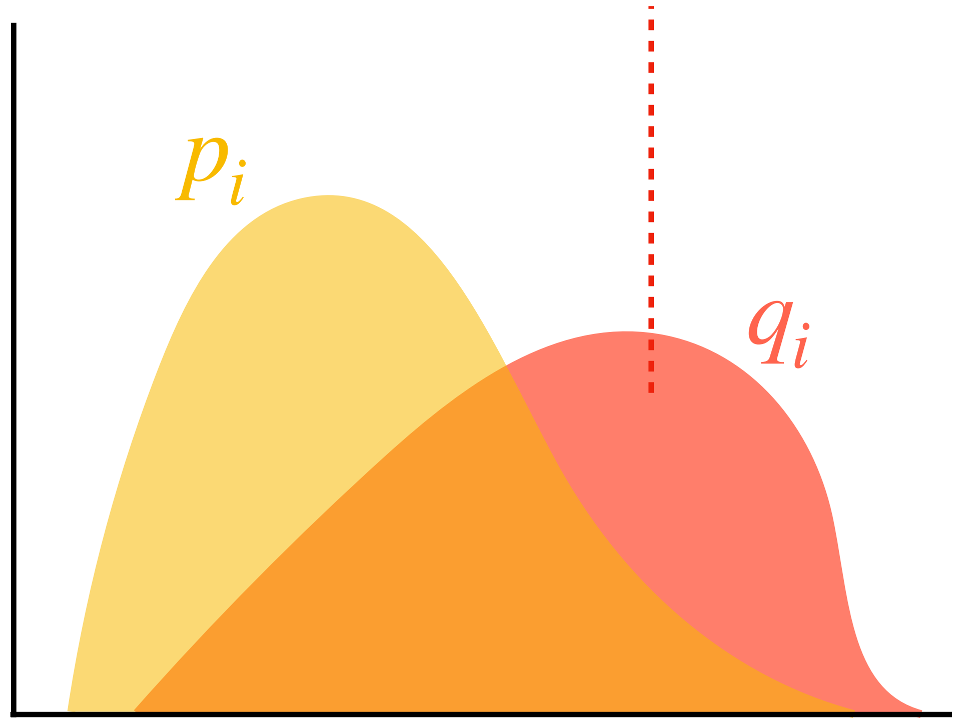
- Case 3: $q_i < p_i$ (reject)

  If $q_i <<< p_i$, we likely would have rejected it.

  In this case, we sample a new token from target model.

  - Specifically, we sample from $(q_i - p_i)_+$

Sample only from this region!



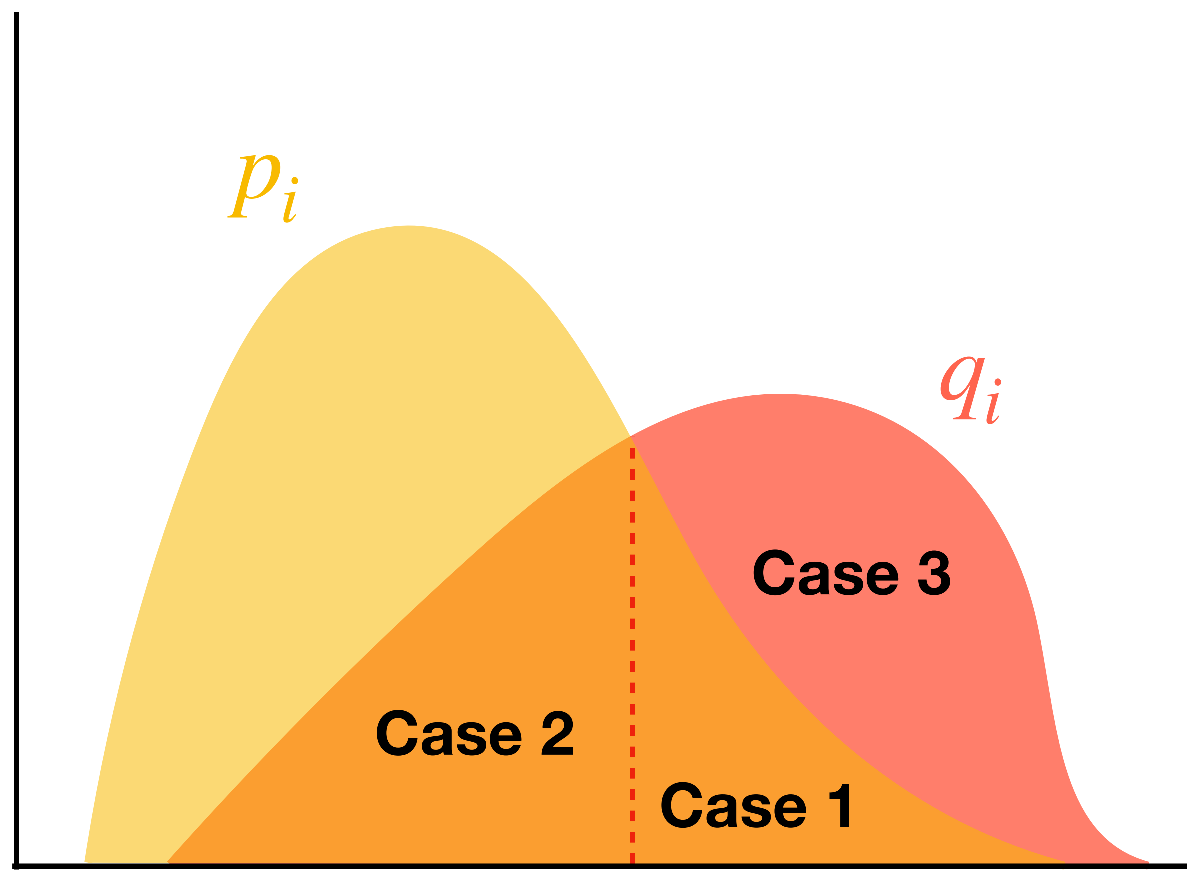
Speeding-up generation: Speculative Sampling

- But why specifically $(q_i - p_i)_+$?
  because our goal: to cover target LM distribution q_i.

- Case 1: $q_i \geq p_i$
  Accept this token.

- Case 2: $q_i < p_i$ (accept)
  Accept it with the probability $\frac{q_i}{p_i}$

- Case 3: $q_i < p_i$ (reject)
  The only remaining case: if token rejected, we sample a new token.
  $(q_i - p_i)_+$ is the only region left to cover q_i!



Note: This sampling procedure, though sampling from small LM (p_i), has the same effect as sampling from target LM (q_i). Formal proof in Appendix I of *(Chen et al., 2023)*

Speeding-up generation: Speculative Sampling

- **Speculative sampling** uses idea of rejection sampling.
  - To sample from a easy-to-sample distribution p (small LM), in order to approximate sampling from a more complex distribution q (large LM).

- Using 4B LM as a draft model and 70B LM as a target model, we get **2~2.5x faster decoding speed** with negligible performance difference!

- Considerations before use
  - M_p and M_q should be pre-trained with the same tokenization scheme!
    (e.g., GPT-2 and GPT- 3 would work, but not GPT-3 and LLaMa-7B)
  - Hardware config matters: If you have 100 GPUs, running large model can actually be faster
    (rather than waiting for a small draft model that only takes up 10 GPU… => *GPU utilization bottleneck*, see page 5-6 in Chen et al.)

Decoding: Takeaways

- Decoding is still a challenging problem in NLG - there's a lot more work to be done!

- Different decoding algorithms can allow us to 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

Training LLMs to avoid exposure bias in generation

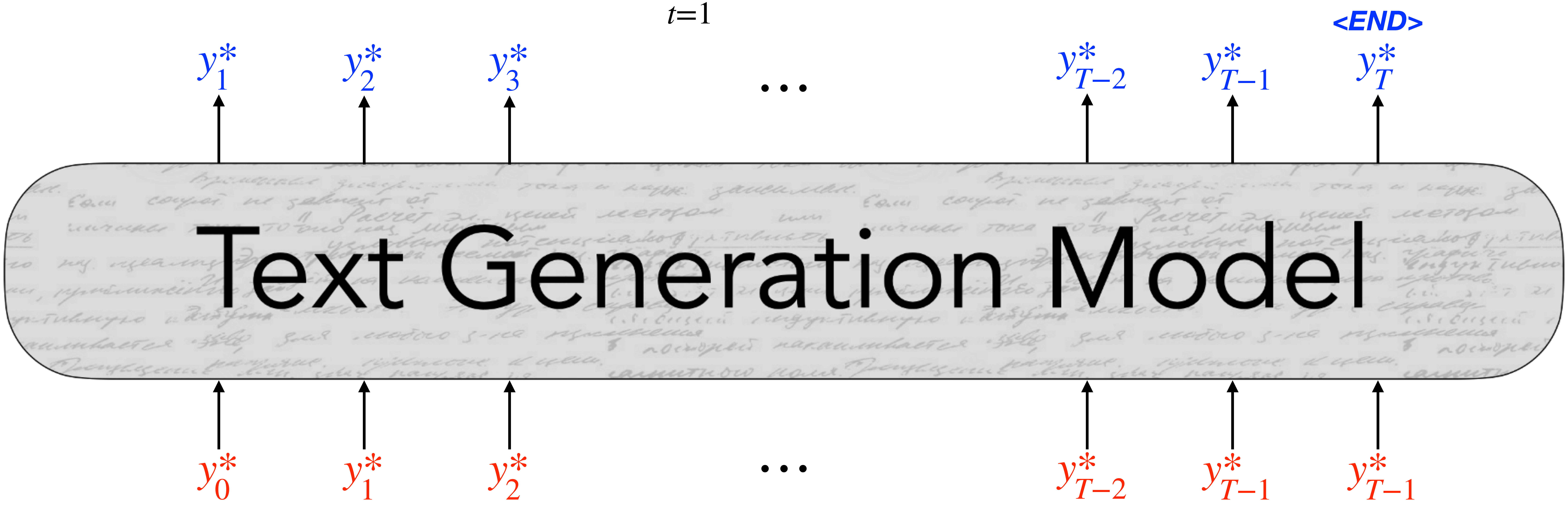
Issues with teacher forcing training of LLMs

- *Teacher forcing* is still the main algorithm for training text generative models
- Diversity is an issue with sequences generated from teacher-forced models
  - Unlikelihood training can help in discouraging undesirable behaviors

Recall: Teacher-forcing

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

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



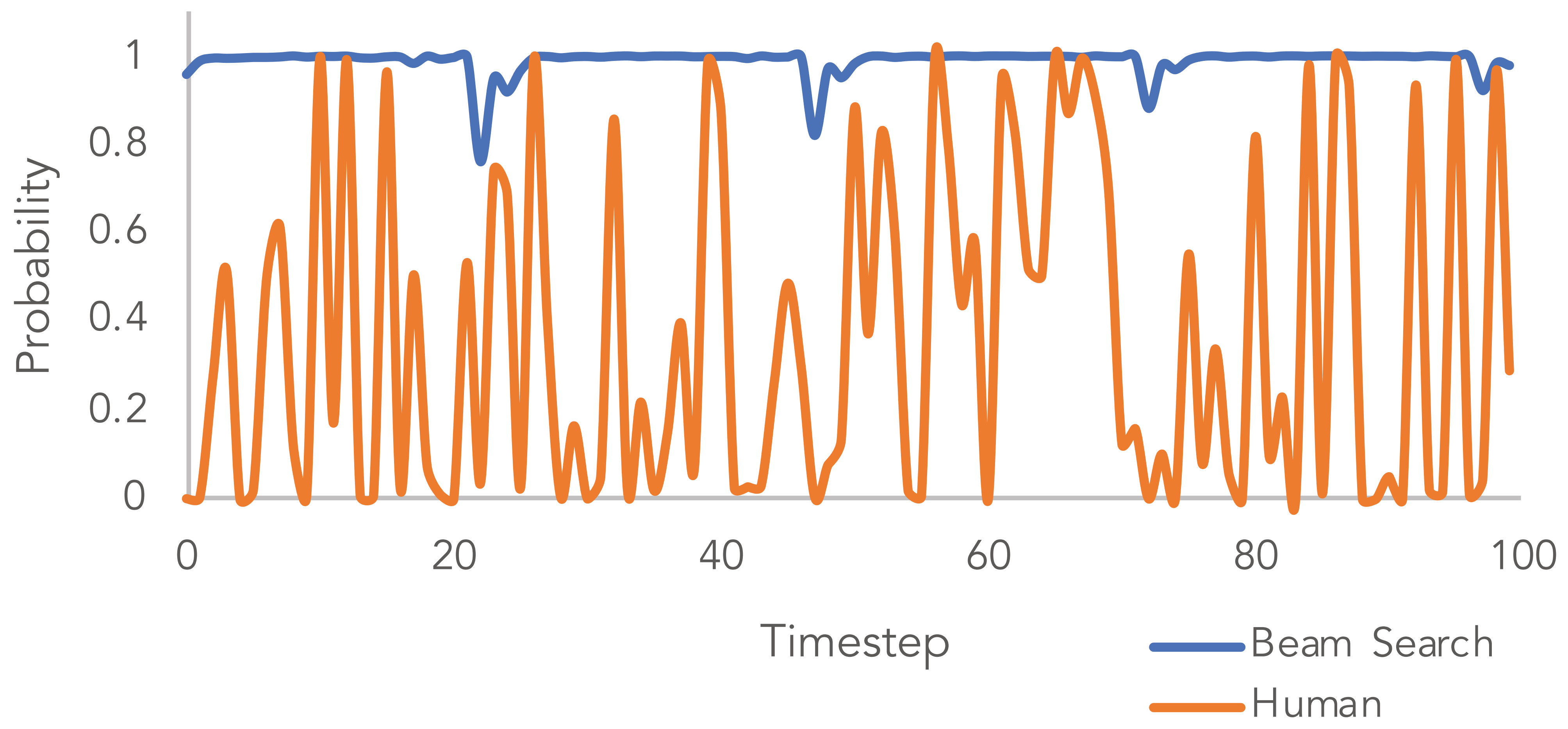
Exposure Bias

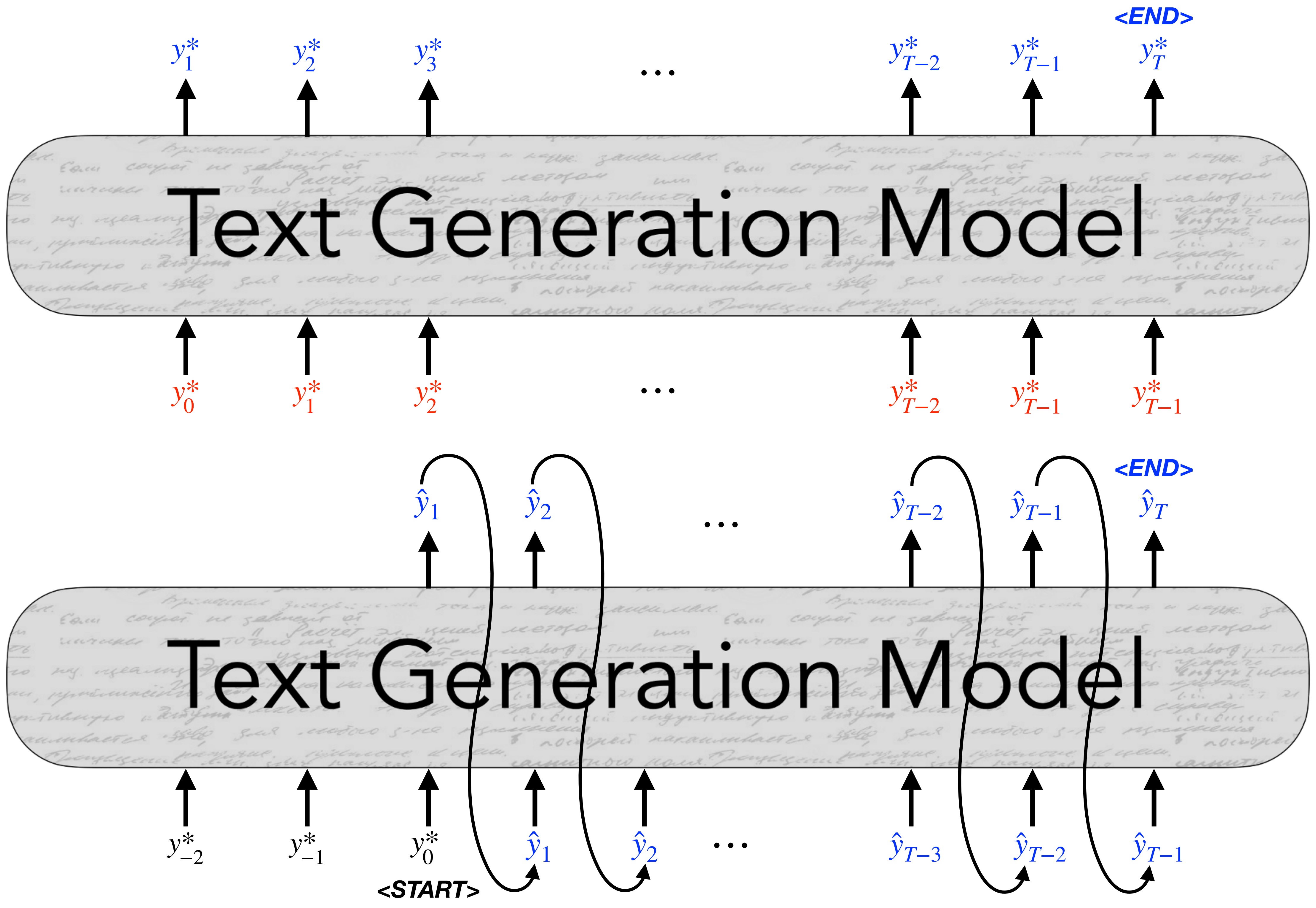
- Training with teacher forcing leads to *exposure bias* at generation time
  - During training, our model's inputs are gold context tokens from real, human-generated texts

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

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

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





y_1^* y_2^* y_3^* … y_{T-2}^* y_{T-1}^* <END> y_T^*
Text Generation Model
y_0^* y_1^* y_2^* … y_{T-2}^* y_{T-1}^* y_{T-1}^*
$\hat{y}_1$ $\hat{y}_2$ … $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ <END> $\hat{y}_T$
Text Generation Model
y_{-2}^* y_{-1}^* y_0^* $\hat{y}_1$ $\hat{y}_2$ … $\hat{y}_{T-3}$ $\hat{y}_{T-2}$ $\hat{y}_{T-1}$
<START>

Exposure Bias Solutions

- Scheduled Sampling *(Bengio et al., 2015)*
  - With some probability p, decode a token and feed that as the next input, rather than the gold token.
  - Increase p over the course of training
  - Leads to improvement in practice, but can lead to a strange training objective

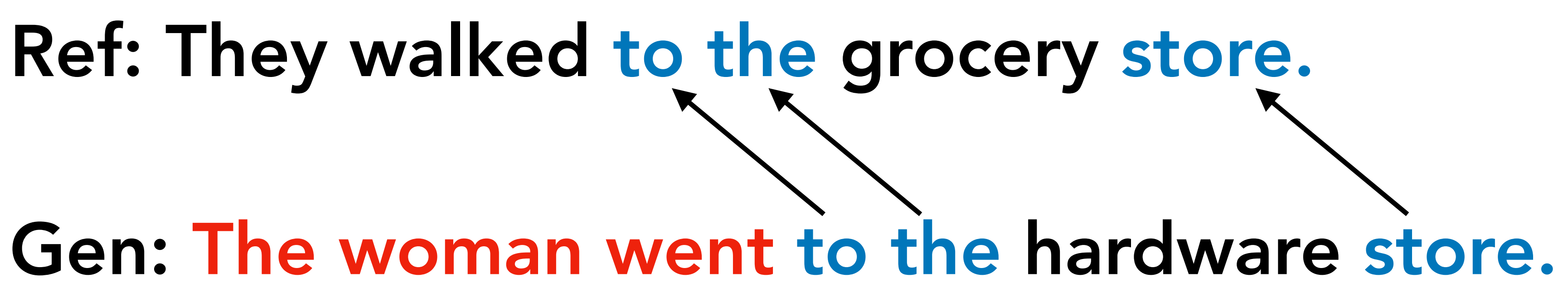
- Dataset Aggregation *(a.k.a. DAgger; Ross et al., 2011)*
- Retrieval Augmentation *(Guu*, Hashimoto* et al., 2018)*
- Reinforcement Learning…

Components of NLG Systems

- What is NLG?
- Formalizing NLG: a simple model and training algorithm
- Decoding from NLG models
- Training NLG models
- **Evaluating NLG Systems**
- Ethical Considerations

Evaluating natural language generation

Types of text evaluation methods



Ref: They walked to the grocery store.

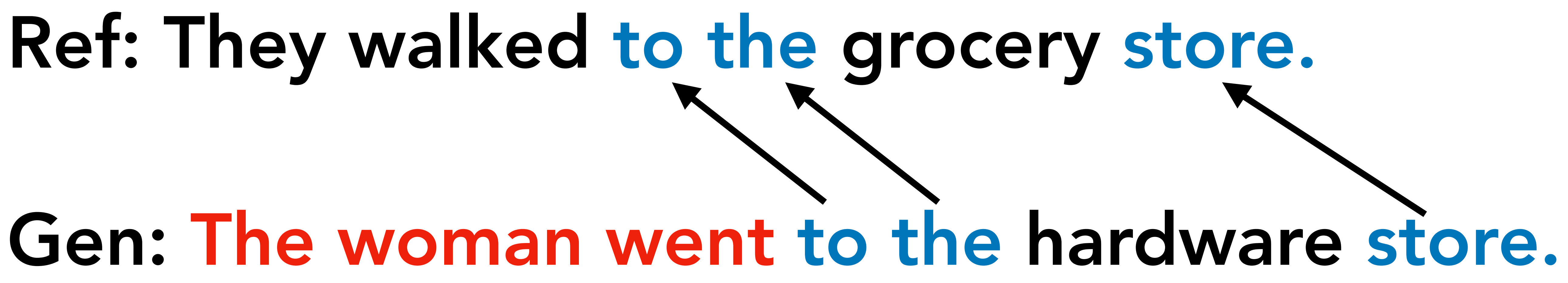
Gen: The woman went to the hardware store.

Content Overlap Metrics

Model-based Metrics

Human Evaluation

Content overlap metrics



Ref: They walked to the grocery store.

Gen: The woman went to the hardware store.

- Compute a score that indicates the similarity between *generated* and *gold-standard* (often human-written) text
- Fast and efficient; widely used (e.g. for MT and summarization)
- Dominant approach: *N*-gram overlap metrics
  - e.g., BLEU, ROUGE, METEOR, CIDEr, etc.

Content overlap metrics

- Dominant approach: *N*-gram overlap metrics
  - e.g., BLEU, ROUGE, METEOR, CIDEr, etc.

- Not ideal even for less open-ended tasks - e.g., machine translation

- They get progressively much worse for more open-ended tasks
  - **Worse** for summarization, as longer summaries are harder to measure
  - **Much worse** for dialogue (in how many ways can you respond to your friend?)
  - **Much, much worse** for story generation, which is also open-ended, but whose sequence length can make it seem you're getting decent scores!

A simple failure case

- *N*-gram overlap metrics have no concept of **semantic relatedness**!

Are you enjoying the NLP class?

For sure!

Score:

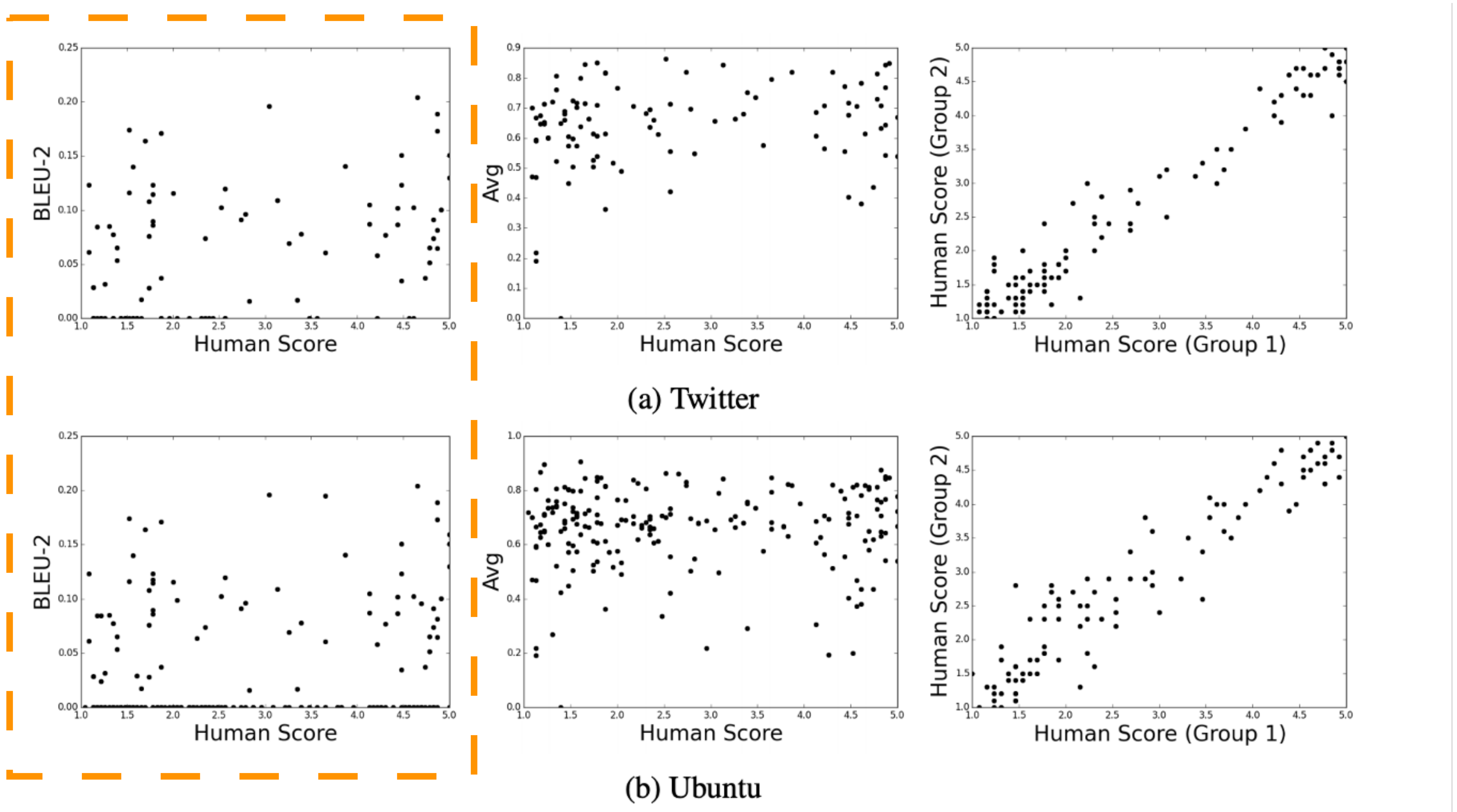
0.61 Yes for sure!

0.25 Sure I do!

False negative 0.0 Yes!

False positive 0.61 No for sure...

A more comprehensive failure analysis



(a) Twitter

(b) Ubuntu

Figure 1: Scatter plots showing the correlation between metrics and human judgements on the Twitter corpus (a) and Ubuntu Dialogue Corpus (b). The plots represent BLEU-2 (left), embedding average (center), and correlation between two randomly selected halves of human respondents (right).

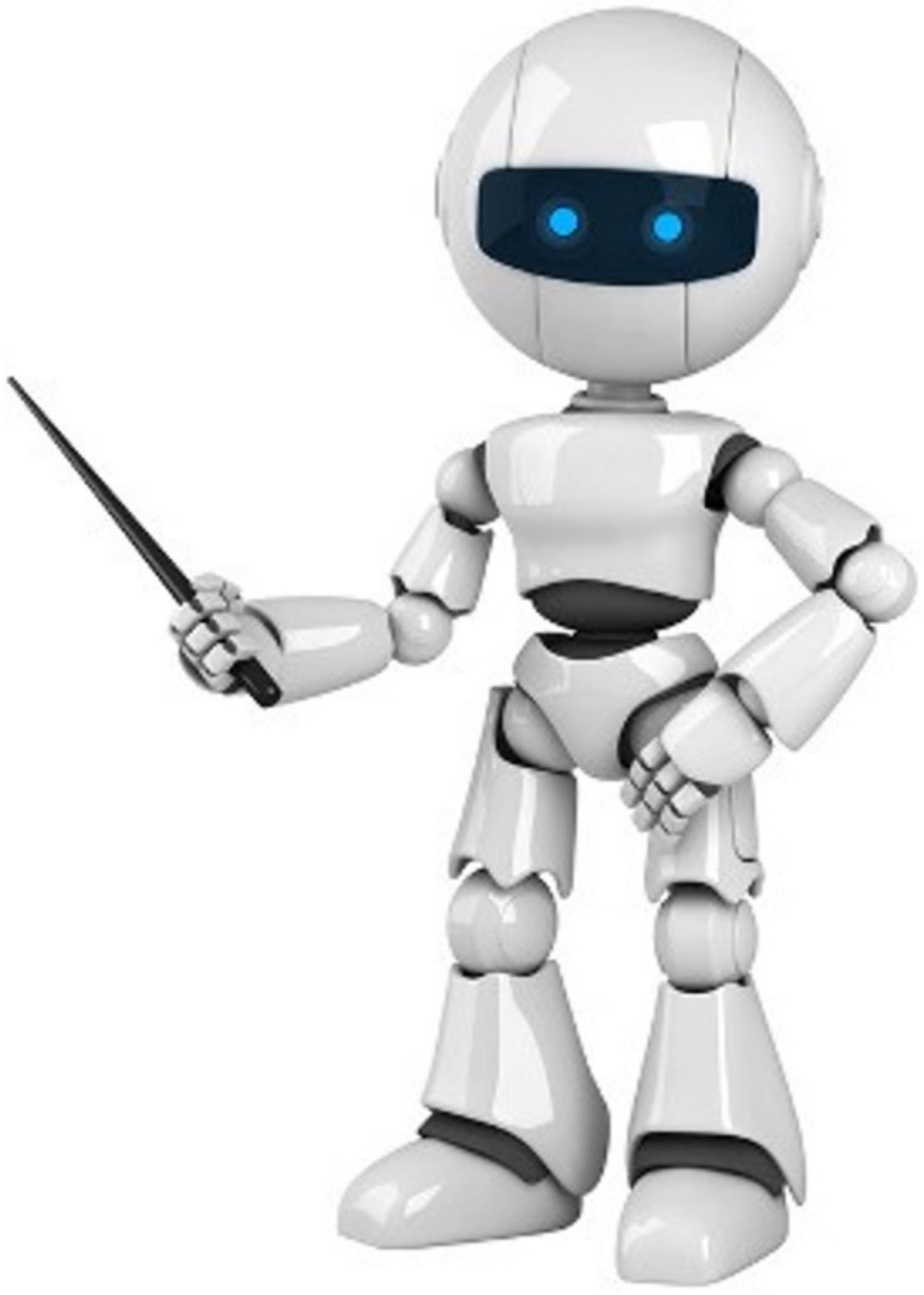
- Higher *n-gram overlap* does not imply higher ***human score***.

Model-based metrics to capture more semantics

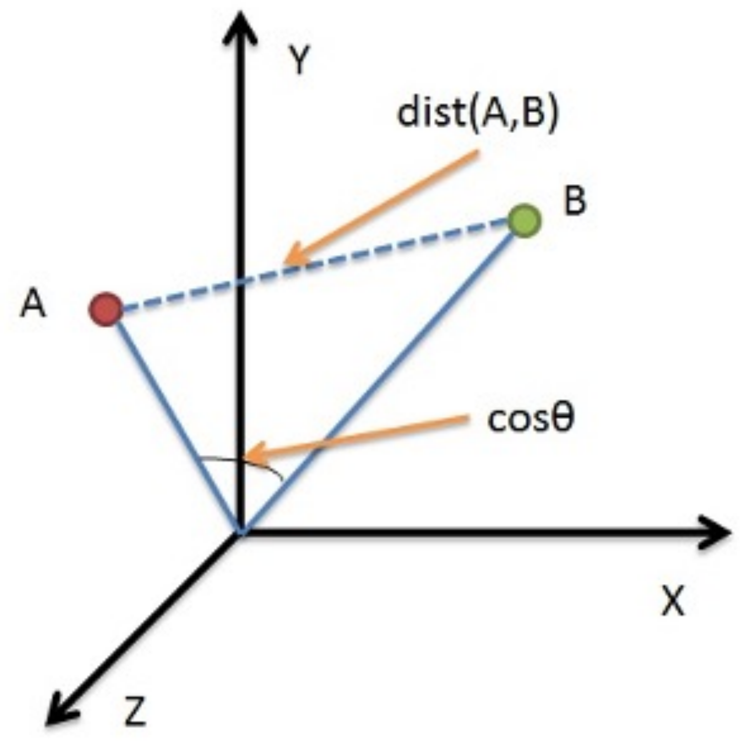
- Use learned representation of words and sentences to compute semantic similarity between generated and reference texts

- No more n-gram bottleneck: text units are represented as embeddings!

- Even though embeddings are pre-trained, distance metrics used to measure similarity can be fixed.



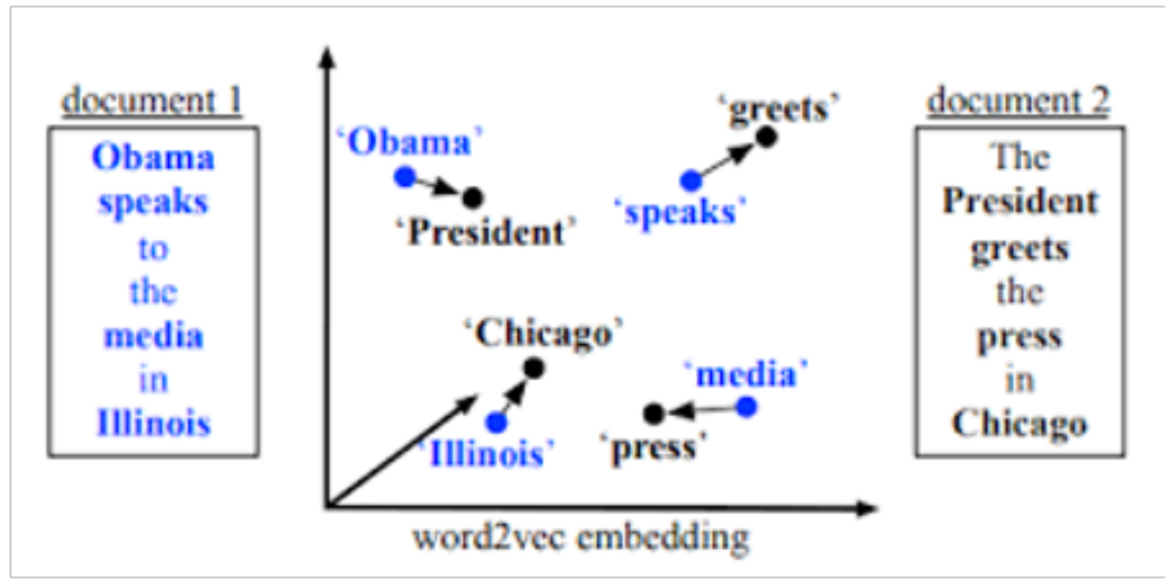
Model-based metrics: Word distance functions



Vector Similarity

Embedding-based similarity for semantic distance between text.

- Embedding Average *(Liu et al., 2016)*
- Vector Extrema *(Liu et al., 2016)*
- MEANT *(Lo, 2017)*
- YISI *(Lo, 2019)*



Word Mover's Distance

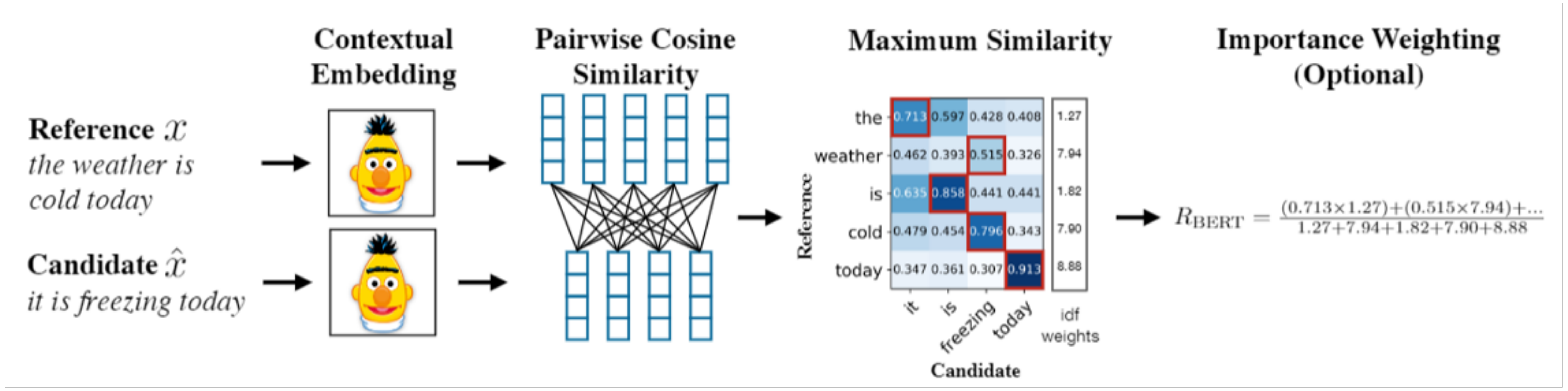
Measures the distance between two sequences using word embedding similarity matching.

- *(Kusner et al., 2015; Zhao et al., 2019)*

BERTSCORE

Uses pre-trained contextual embeddings from BERT and matches words in candidate and reference sentences by cosine similarity.

- *(Zhang et al., 2019)*

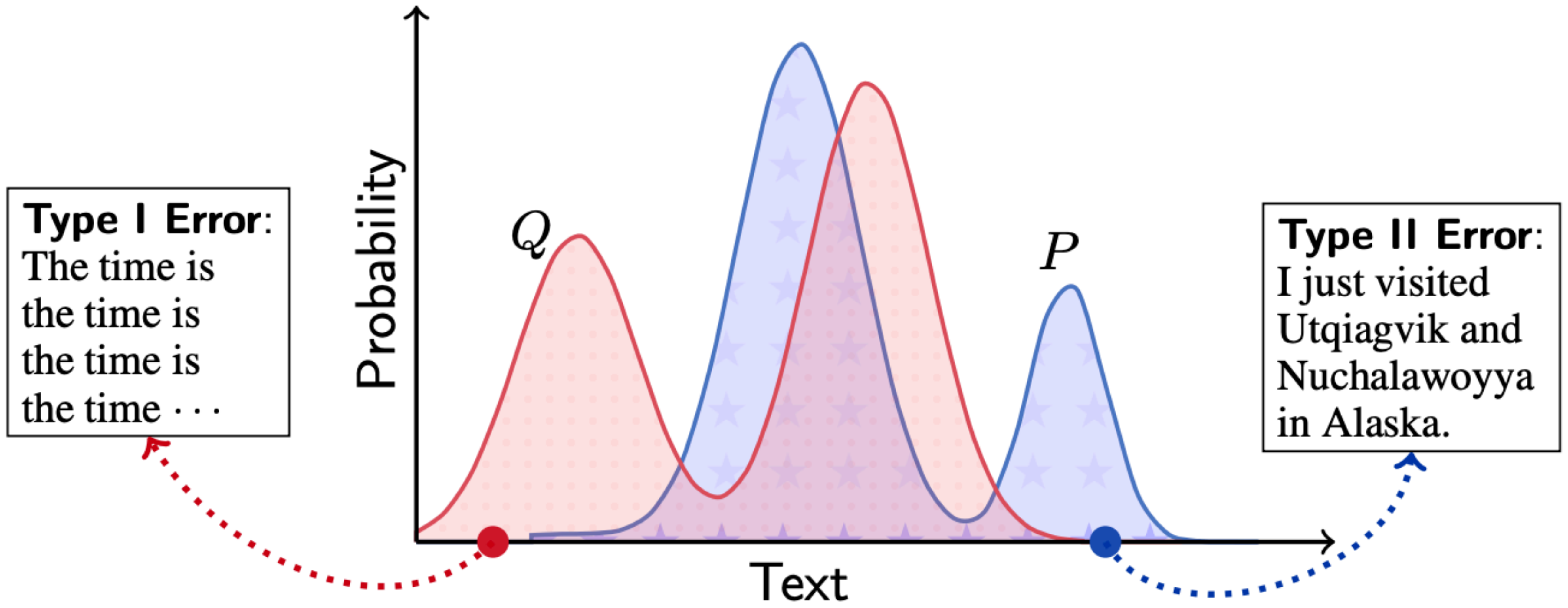


Reference	it	is	freezing	today	idf weights
the	0.713	0.597	0.428	0.408	1.27
weather	0.462	0.393	0.515	0.326	7.94
is	0.635	0.858	0.441	0.441	1.82
cold	0.479	0.454	0.796	0.343	7.90
today	0.347	0.361	0.307	0.913	8.88

$$R_{\text{BERT}} = \frac{(0.713\times1.27)+(0.515\times7.94)+\ldots}{1.27+7.94+1.82+7.90+8.88}$$

MAUVE: Beyond single sample matching

- In open-ended generation, comparing with a single reference may not say much.
  Can we instead compare the distribution of machine text vs. human text?
- **MAUVE** *(Pillutla et al., 2021)*
  - Computes the information divergence between the human text distribution P and the machine text distribution Q



Type I Error: The time is the time is the time is the time · · ·

Type II Error: I just visited Utqiagvik and Nuchalawoyya in Alaska.

MAUVE: Beyond single sample matching

- Divergence Curve

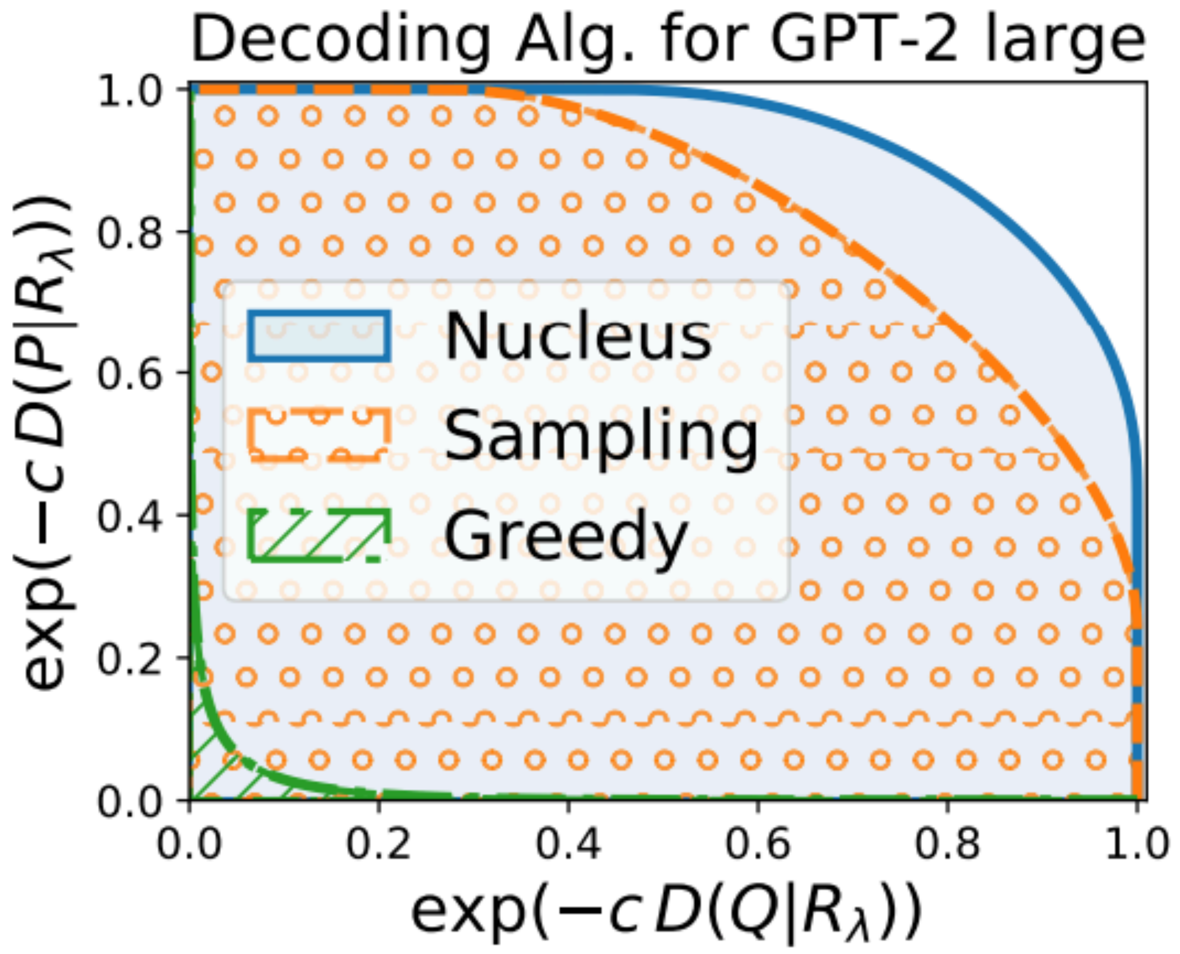
$$\mathcal{C}(P,Q) = \Big\{ \big(\exp(-c\,\underline{\mathrm{KL}(Q|R_\lambda)}), \exp(-c\,\mathrm{KL}(P|R_\lambda)) \big) \;:\; \underline{R_\lambda = \lambda P + (1-\lambda)Q,}\; \lambda \in (0,1) \Big\}$$

KL Divergence: Distance between two distributions Q and R_λ

Interpolate between P and Q to draw a curve

$$\mathrm{KL}(P|R_\lambda) = \sum_{\boldsymbol{x}} P(\boldsymbol{x}) \log \frac{P(\boldsymbol{x})}{R_\lambda(\boldsymbol{x})}$$

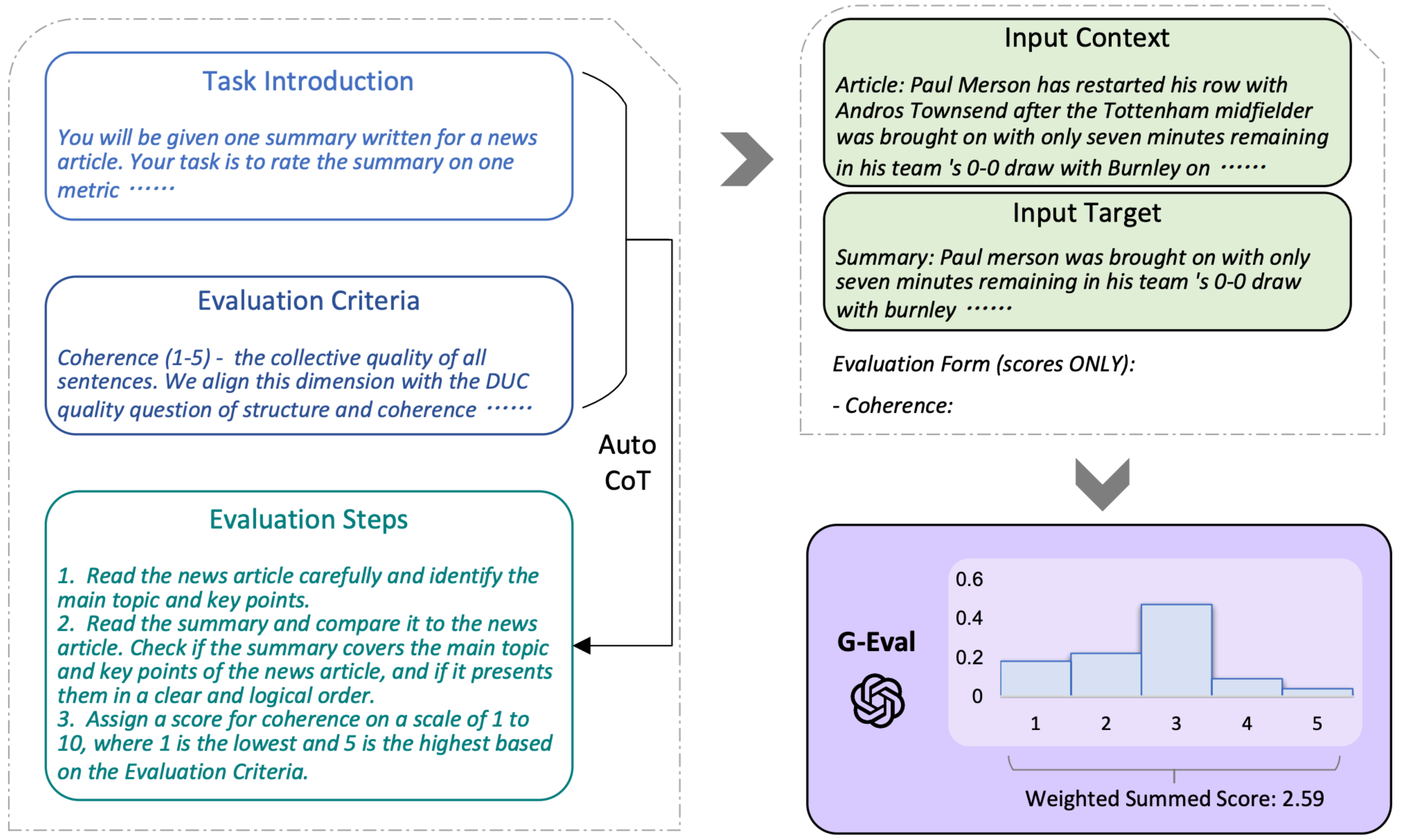
- If P and Q are close, KL divergence will be lower, thus the divergence curve will be higher
- **MAUVE(P, Q)**: Area under the divergence curve (value in 0~1, **higher is better!**)



Nucleus sampling is better than naive sampling / greedy decoding.

Model-based metrics: LLM as evaluator

- Directly prompt LLM (GPT-4) to evaluate generated text.
  - Can be customized with evaluation criteria
  - (Often) better correlation with human evaluators than task-specific metrics (e.g. ROUGE)
  - (Often) is cheaper than human evaluation



Task Introduction

You will be given one summary written for a news article. Your task is to rate the summary on one metric ……

Evaluation Criteria

Coherence (1-5) - the collective quality of all sentences. We align this dimension with the DUC quality question of structure and coherence ……

Evaluation Steps

1. Read the news article carefully and identify the main topic and key points.
2. Read the summary and compare it to the news article. Check if the summary covers the main topic and key points of the news article, and if it presents them in a clear and logical order.
3. Assign a score for coherence on a scale of 1 to 10, where 1 is the lowest and 5 is the highest based on the Evaluation Criteria.

Input Context

Article: Paul Merson has restarted his row with Andros Townsend after the Tottenham midfielder was brought on with only seven minutes remaining in his team 's 0-0 draw with Burnley on ……

Input Target

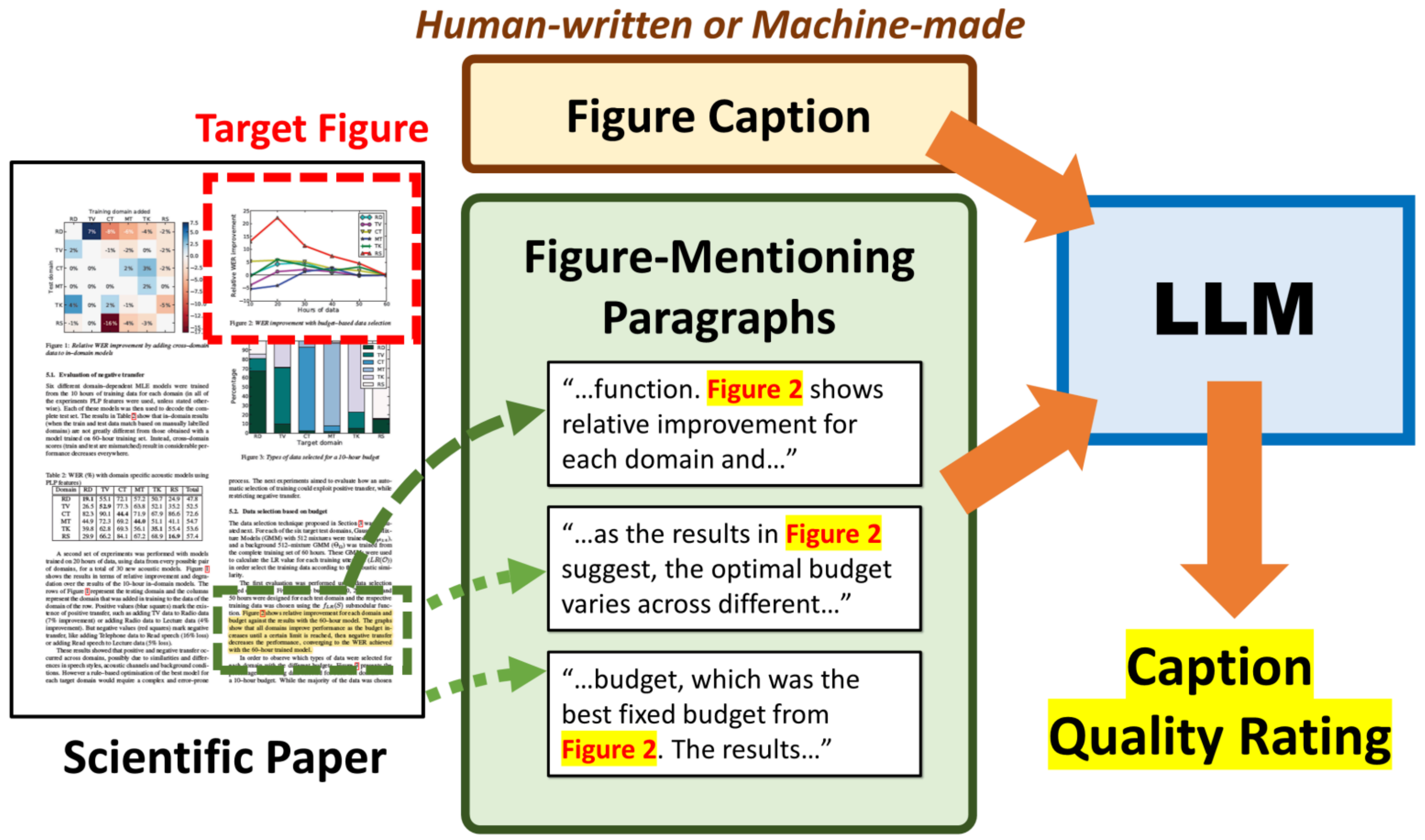
Summary: Paul merson was brought on with only seven minutes remaining in his team 's 0-0 draw with burnley ……

Evaluation Form (scores ONLY):

- Coherence:

Liu et al. 2023

- Limitations
  - Brittleness: LLM evaluation can significantly vary when given different prompts!
  - Potential self-bias - LLMs may prefer what LLMs have generated…



“…function. Figure 2 shows relative improvement for each domain and…”

“…as the results in Figure 2 suggest, the optimal budget varies across different…”

“…budget, which was the best fixed budget from Figure 2. The results…”

Hsu et al. EMNLP Findings, 2023

Human evaluations

- Automatic metrics fall short of matching human decisions
- Most important form of evaluation for text generation systems
- Gold standard in developing new automatic metrics
  - Better automatic metrics will better correlate with human judgements!

Human evaluations

- Sounds easy, but hard in practice: Ask humans to evaluate the quality of text

- Typical evaluation dimensions:
  - fluency
  - coherence / consistency
  - factuality and correctness
  - commonsense
  - style / formality
  - grammaticality
  - typicality
  - redundancy
  - ...

Note: Don't compare human evaluation scores across different studies

Even if they claim to evaluate on the same dimensions!

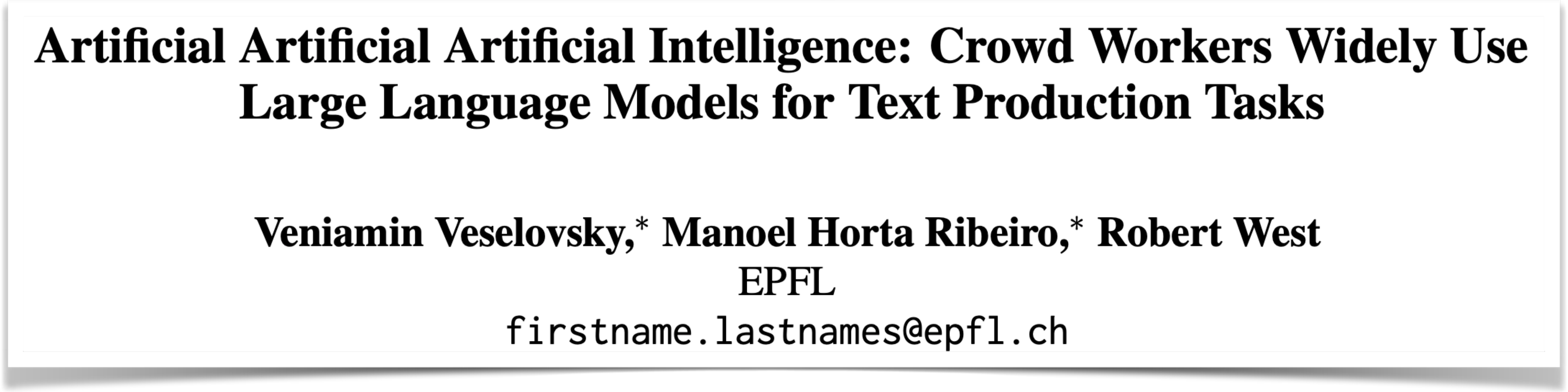
Human evaluations

- Human judgments are regarded as **gold standard**
- Of course, we know that human eval is slow and expensive
- Beyond its cost, human eval is still far from perfect:

- Human judgements
  - are inconsistent / irreproducible
  - can be illogical
  - can be misinterpreting your questionnaire
  - …

  - and recently, use of LLMs by crowd-source workers 🙄

(Veselovsky et al., 2023)



Artificial Artificial Artificial Intelligence: Crowd Workers Widely Use Large Language Models for Text Production Tasks

Veniamin Veselovsky,* Manoel Horta Ribeiro,* Robert West
EPFL
`firstname.lastnames@epfl.ch`

Evaluation: Takeaways

- *Content-overlap metrics* provide a good starting point for evaluating the generation quality, but they're not good enough on their own

- *Model-based metrics* can be more correlated with human judgment, but often are not interpretable

- Human judgments are critical
  - But humans are inconsistent!

- In many cases, the best judge of output quality is **YOU**!
  - **Look at the actual generations - don't just rely on numbers.**
  - **Publicly release large samples of outputs from your system!**

Concluding Thoughts

- Interacting with NLG systems quickly shows their limitations

- Even in tasks with more progress, there are still many improvements ahead

- Evaluation remains a huge challenge
  - We need betters ways to automatically evaluate NLG systems

- One of the most exciting areas of NLP to work in!