COMP 3361 Natural Language Processing

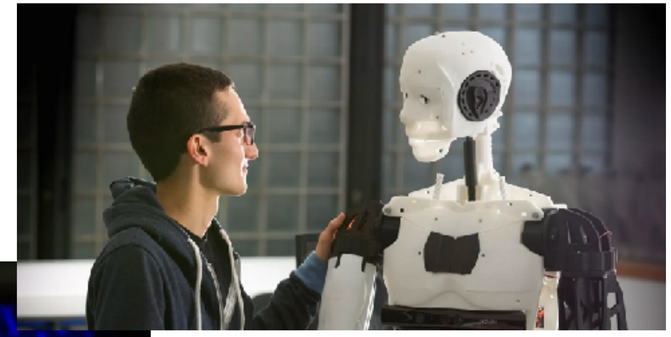
Lecture 16: Natural language generation with LLMs

Spring 2025

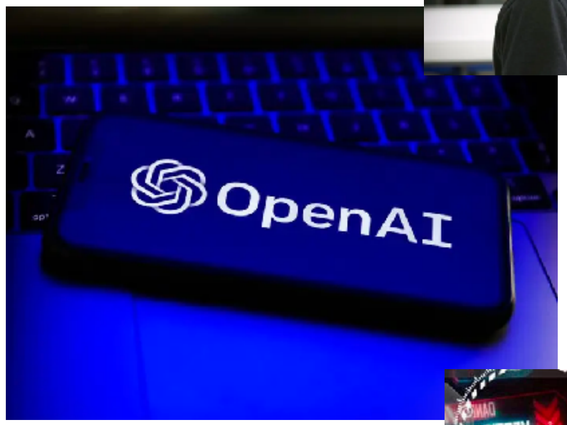
Many materials from CSE447@UW (Jaehun Jung) with special thanks!

Natural language generation with LLMs

- Large Language Models are (mostly) **natural language generation (NLG)** systems!
- The process of generating natural language text 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**!

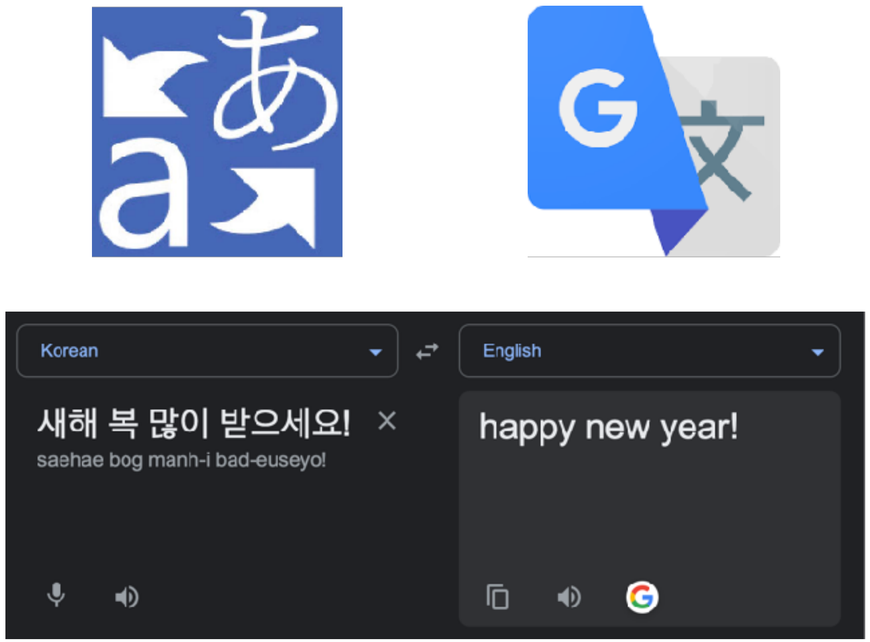


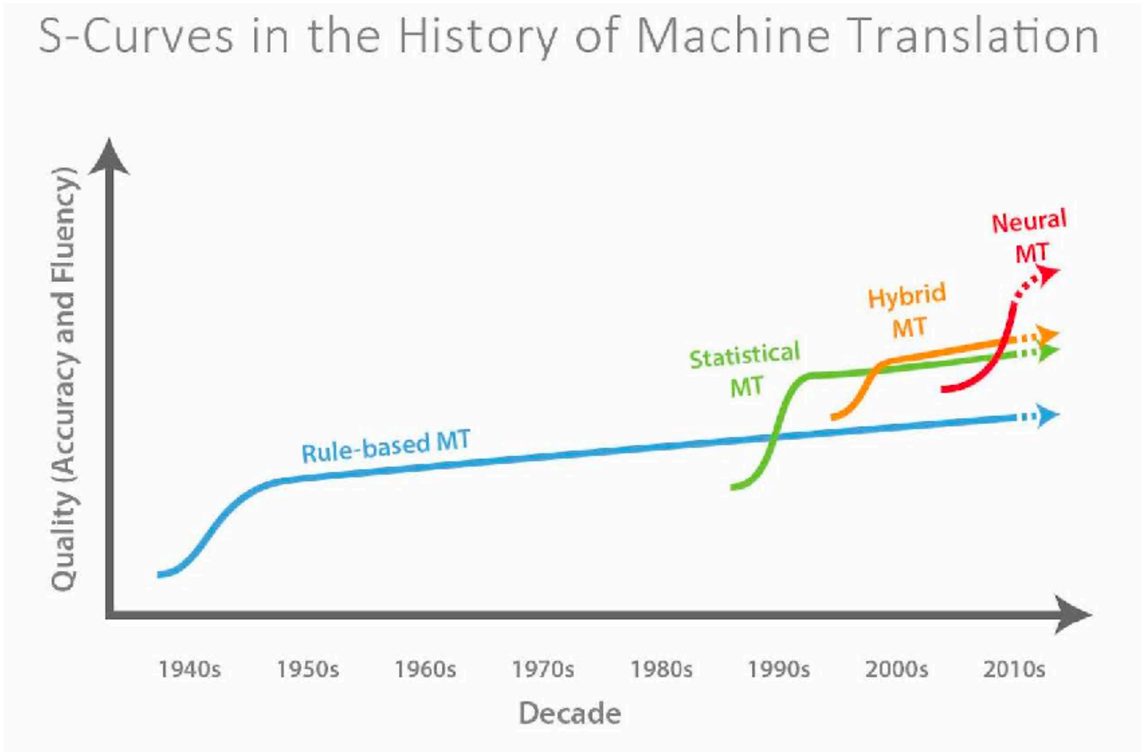
© University of Lincoln



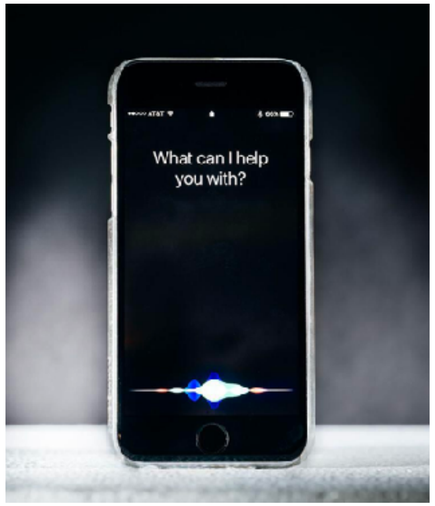
© Marvel Studios

Machine translation



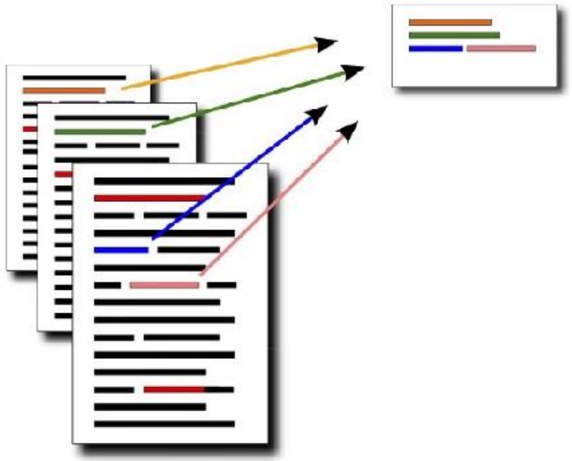


Dialogue systems



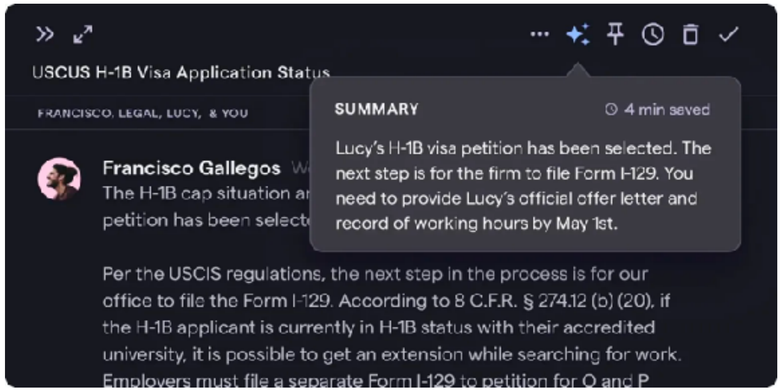
Summarization

Document Summarization



© http://mogren.one/lic/

Email Summarization



© techcrunch.com

Meeting Summarization

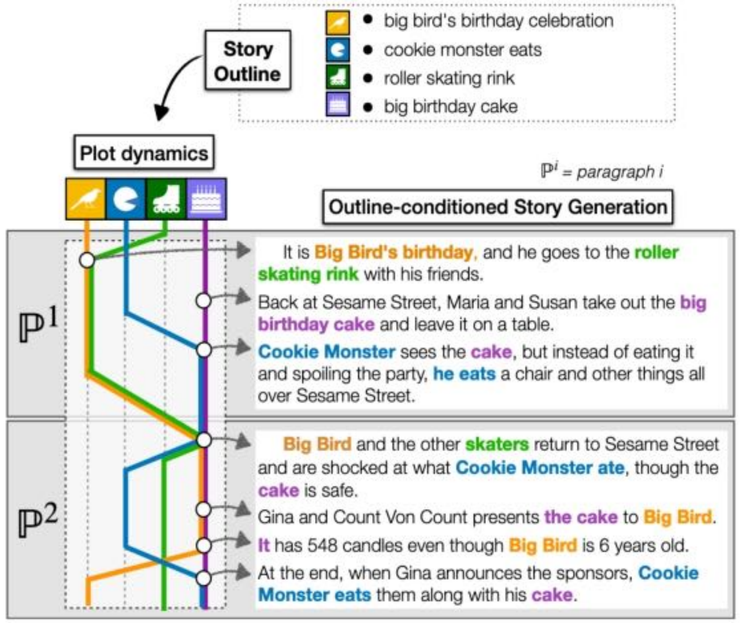
Speaker 1: We'll do it on 18 is fine.
Speaker 4: Okay
Speaker 7: Alex Vasquez will get the step forward.
Speaker 0: Good evening, Mayor and city council. I'm going to turn it over to Jolene Richardson.
Speaker 1: She's our risk manager and she'll give a brief overview of this particular report. Even 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

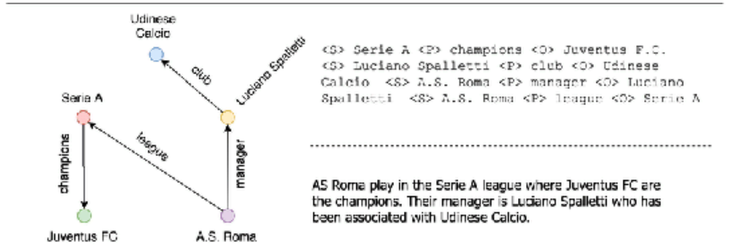
More interesting NLG uses

Creative Stories



Rashkin et al., EMNLP 2020

Data-to-Text Generation



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

Even more NLG tasks

Code generation

```python
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]
```

```python
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

Math reasoning

Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?
Solution: Beth bakes 4 2 dozen batches of cookies for a total of 4*2 = <<4*2=8>>8 dozen cookies
There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of 12*8 = <<12*8=96>>96 cookies
She splits the 96 cookies equally amongst 16 people so they each eat 96/16 = <<96/16=6>>6 cookies
Final Answer: 6

Problem: Mrs. Lim milks her cows twice a day. Yesterday morning, she got 68 gallons of milk and in the evening, she got 82 gallons. This morning, she got 18 gallons fewer than she had yesterday morning. After selling some gallons of milk in the afternoon, Mrs. Lim has only 24 gallons left. How much was her revenue for the milk if each gallon costs $3.50?
Mrs. Lim got 68 gallons - 18 gallons = <<68-18=50>>50 gallons this morning.
So she was able to get a total of 68 gallons + 82 gallons + 50 gallons = <<68+82+50=200>>200 gallons.
She was able to sell 200 gallons - 24 gallons = <<200-24=176>>176 gallons.
Thus, her total revenue for the milk is $3.50/gallon x 176 gallons = $<<3.50*176=616>>616.
Final Answer: 616

More…

Categorization of NLG tasks

Spectrum of open-endedness for NLG tasks

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

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

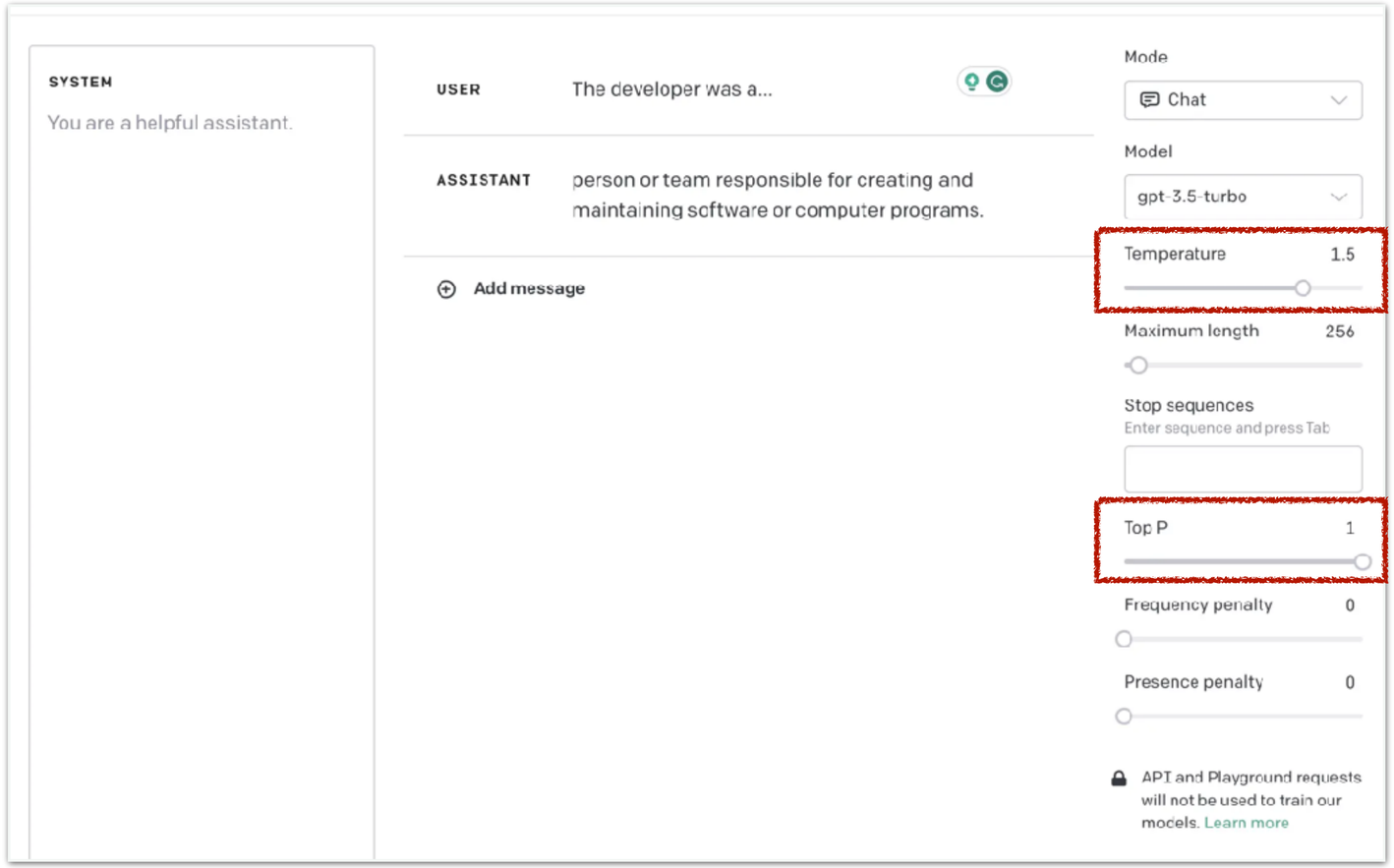
Less open-ended ***More open-ended***

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!

How to control open-endedness in ChatGPT?

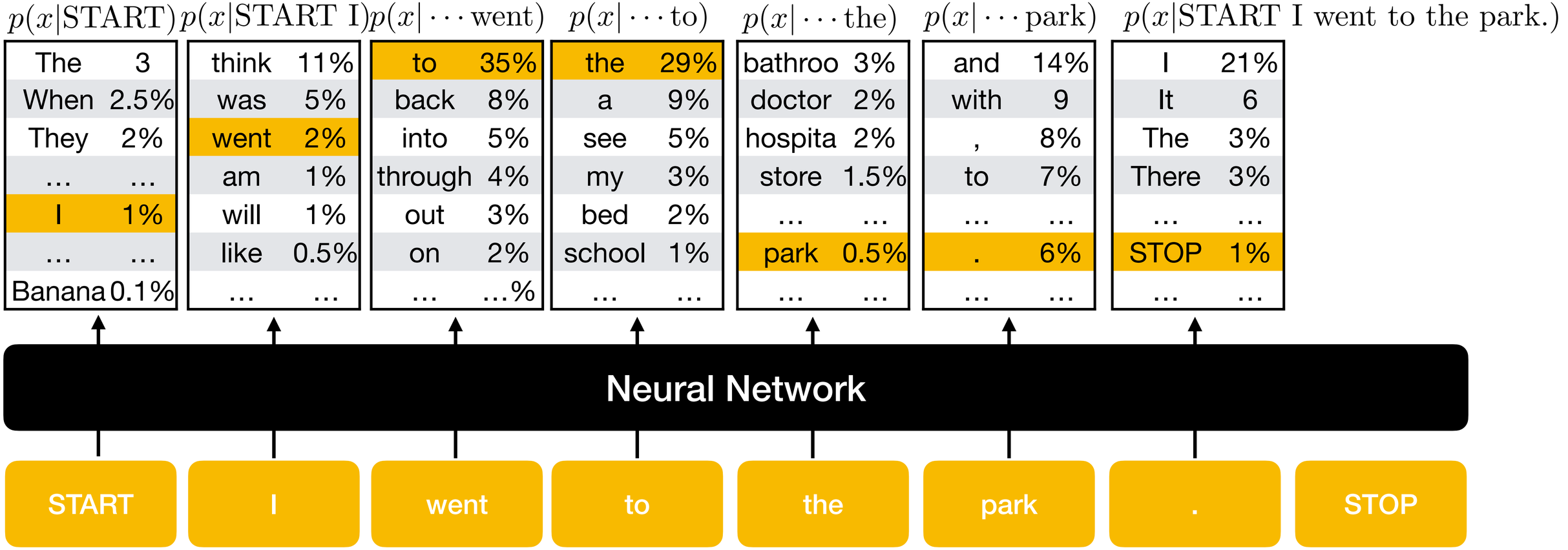


ChatGPT API web interface

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

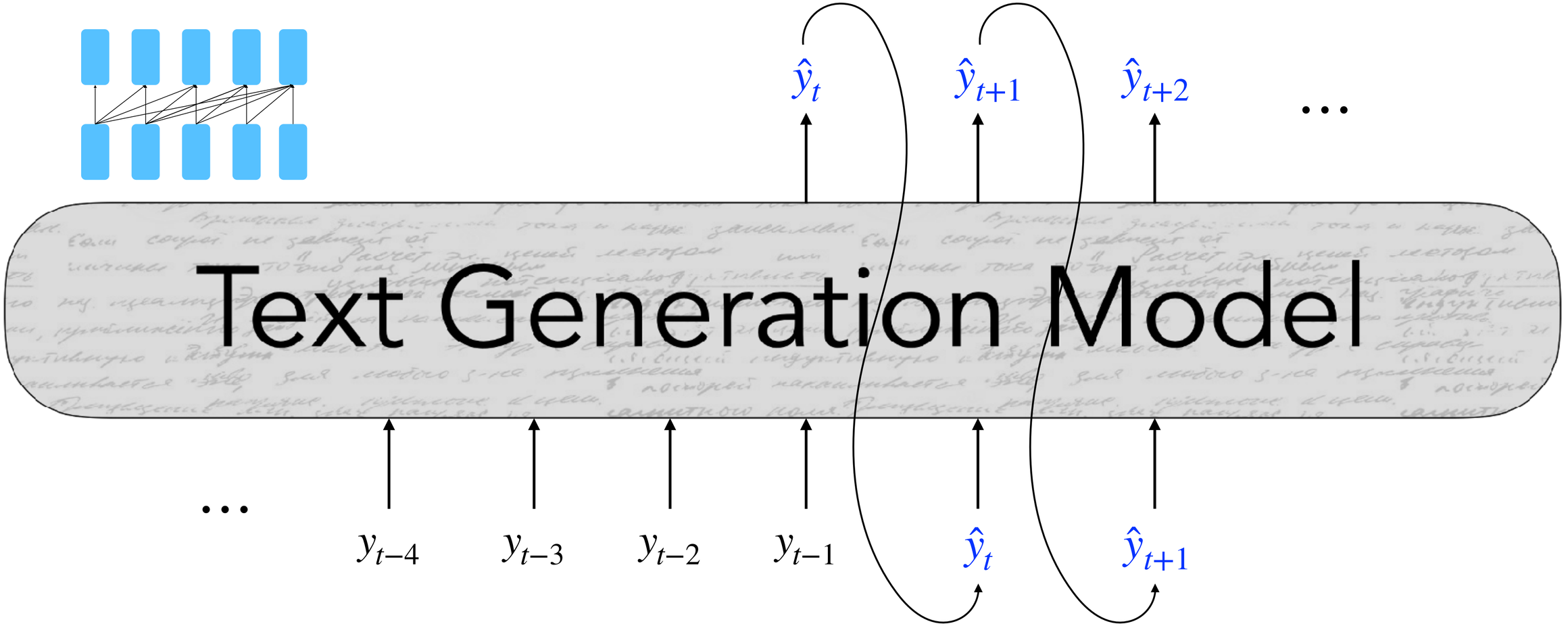
Neural language models

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



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, $\hat{y}_t$



Autoregressive NLG with LLMs

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

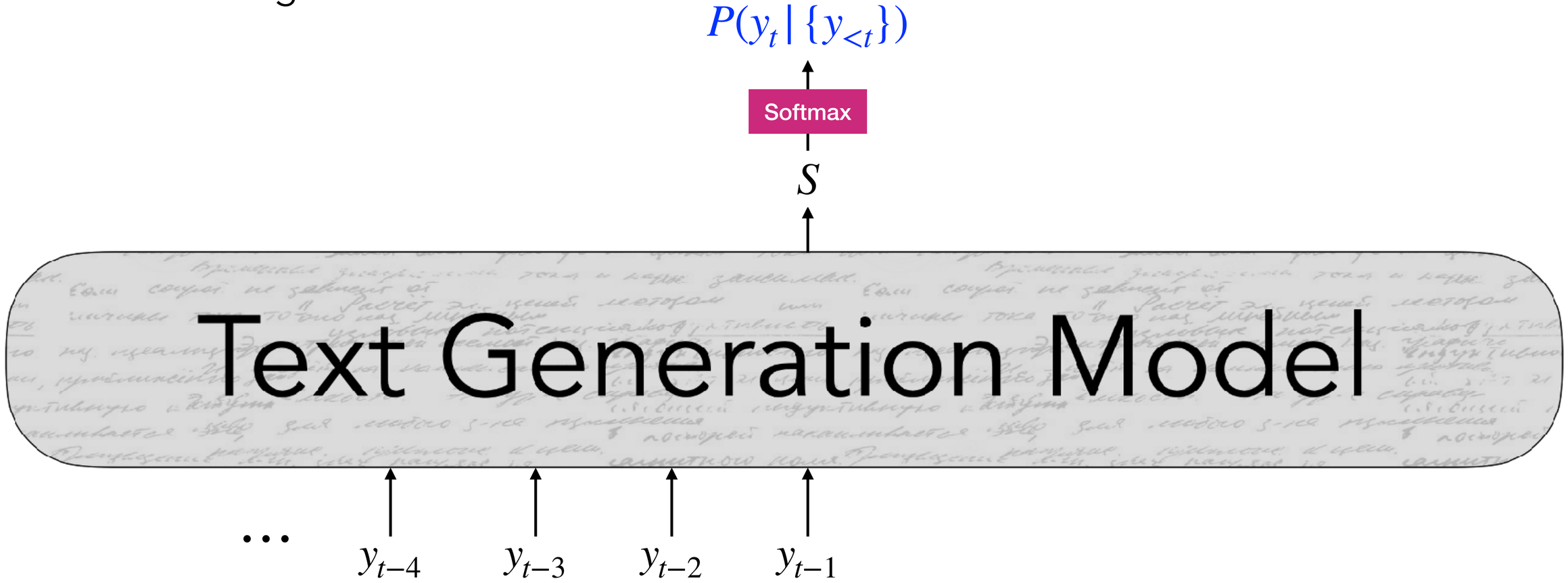
$$S = \underbrace{f(\{y_{<t}\}; \theta)}_{f(\cdot\,; \theta) \text{ is your model}}$$

- 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:

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

$g(\cdot)$ 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:

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

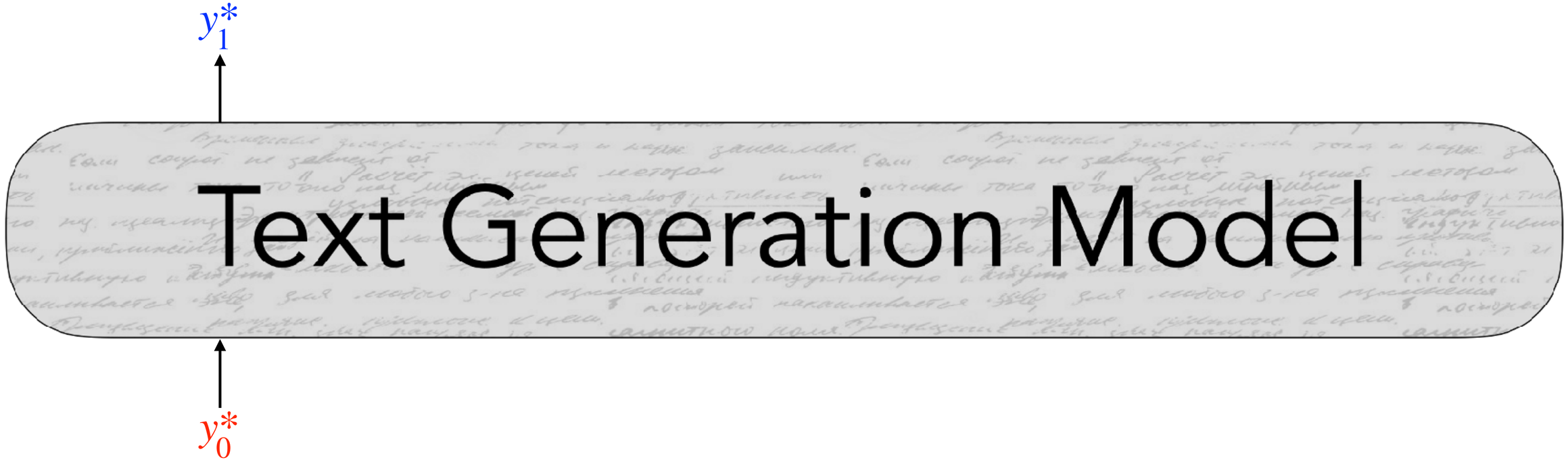
Remark:

- 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 $\{y^*\}_{<t}$

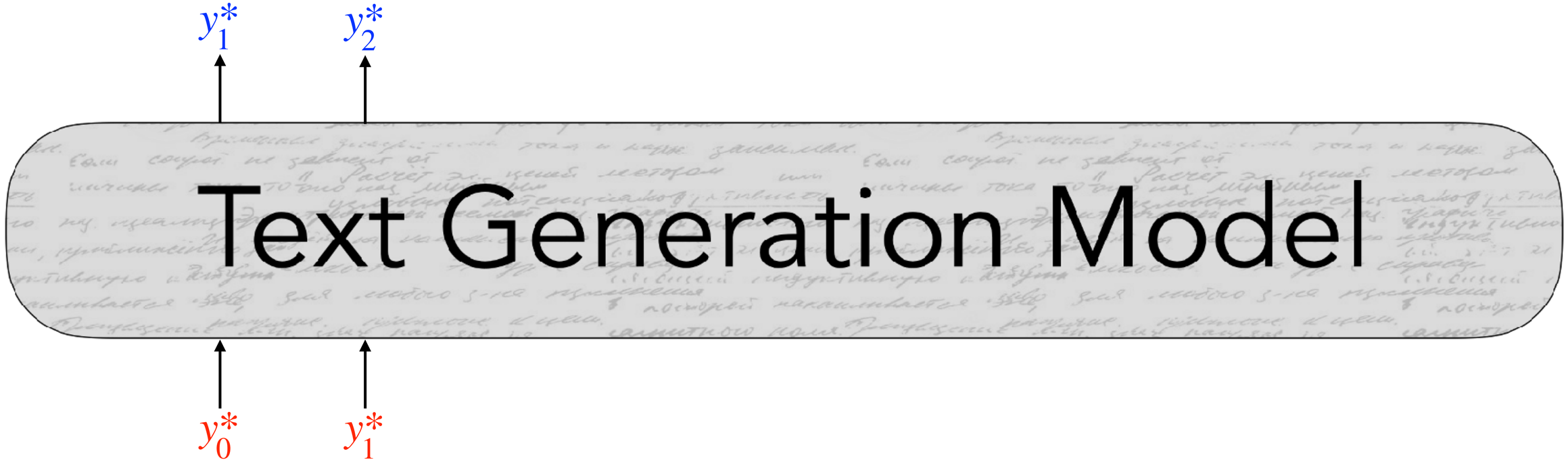
$$L = -\log P(y_1^* \mid y_0^*)$$



Recap: Maximum Likelihood Training

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

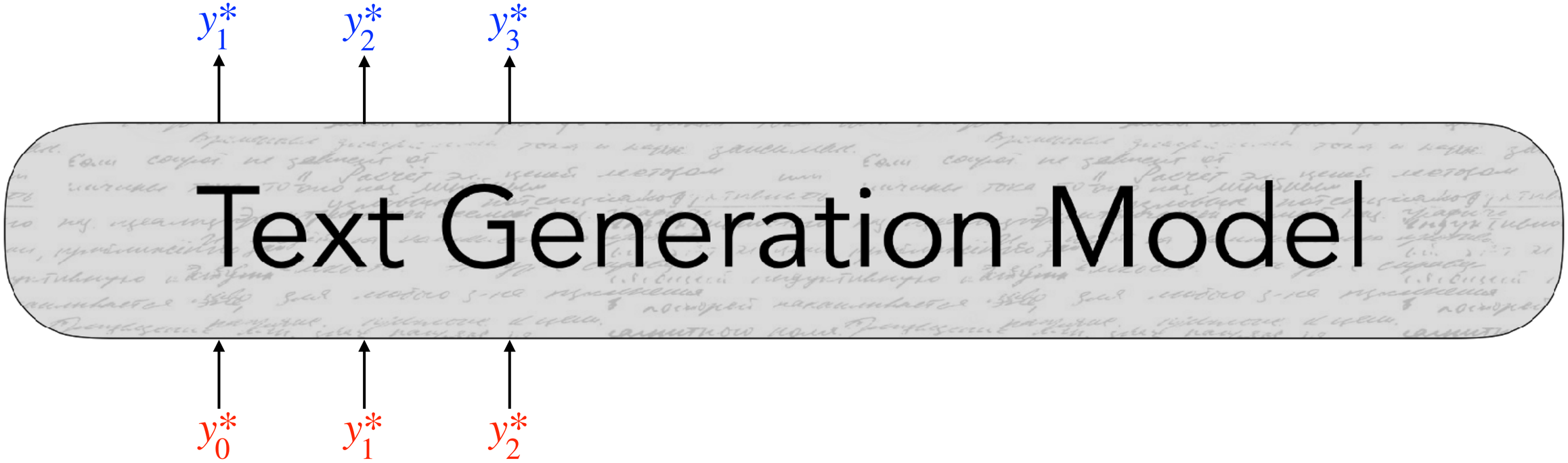
$$L = -\left(\log P(y_1^* \mid y_0^*) + \log P(y_2^* \mid y_0^*, y_1^*)\right)$$



Recap: Maximum Likelihood Training

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

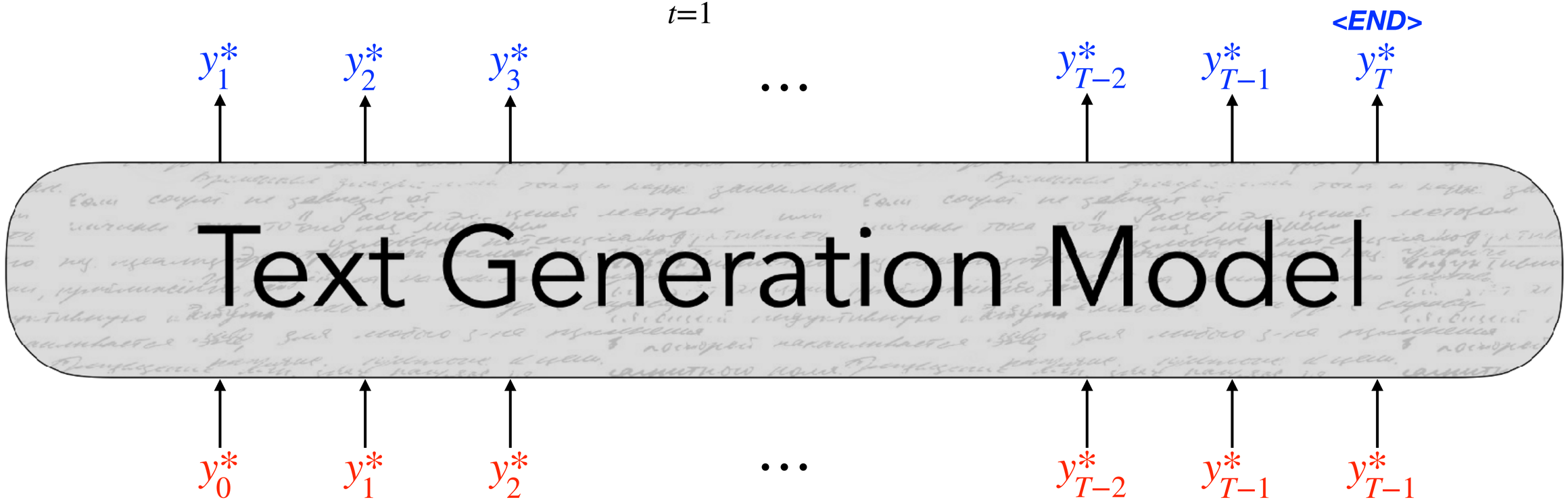
$$L = -\left(\log P(y_1^* \mid y_0^*) + \log P(y_2^* \mid y_0^*, y_1^*) + \log P(y_3^* \mid y_0^*, y_1^*, y_2^*)\right)$$



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, $S \in \mathbb{R}^{V}$:

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

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

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

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

$$\hat{y}_t = \underbrace{g(P(y_t \mid \{y_{<t}\}))}_{g(\cdot) \text{ is your decoding algorithm}}$$

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

How to find the most likely text to generate?

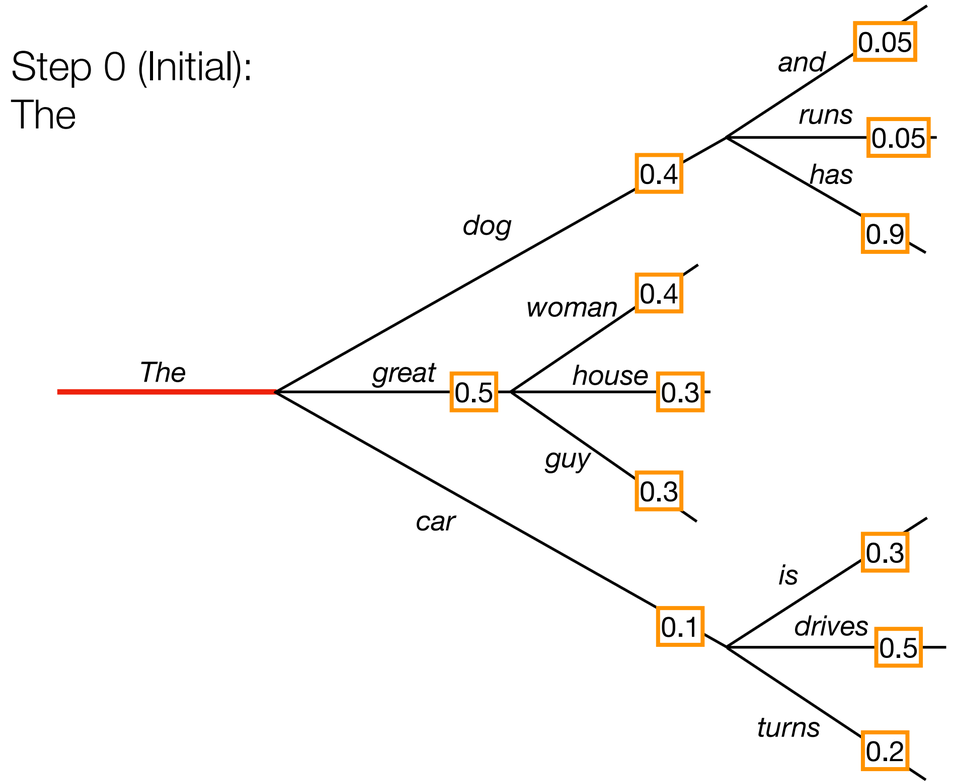
- **Obvious method: Greedy Decoding**
  - Selects the highest probability token according to $P(y_t \mid y_{<t})$

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

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

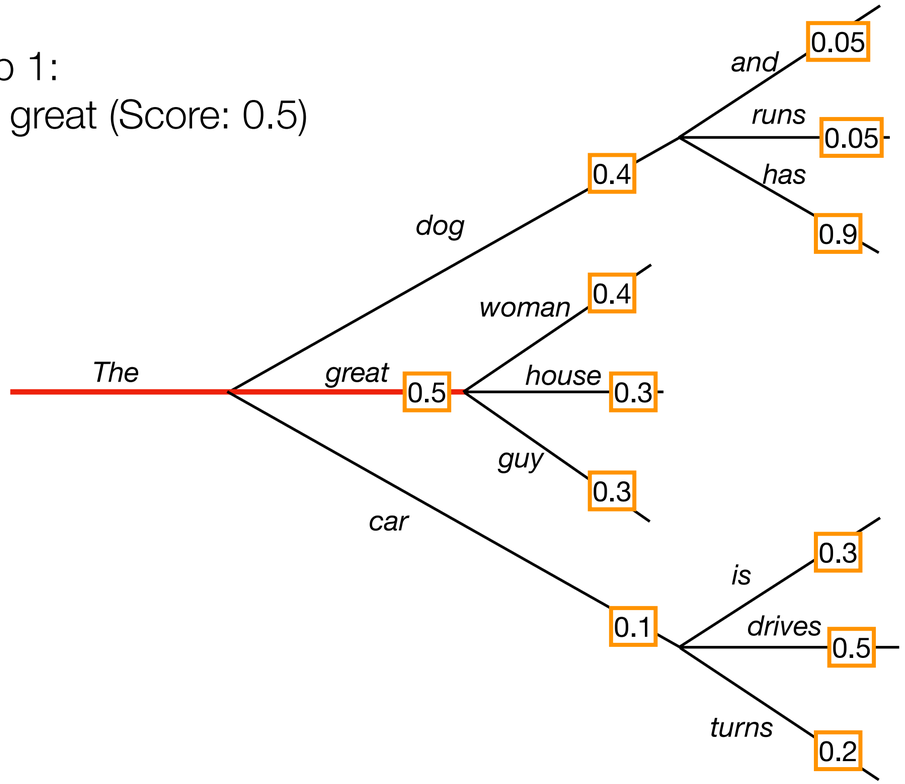
Greedy Decoding vs. Beam Search

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



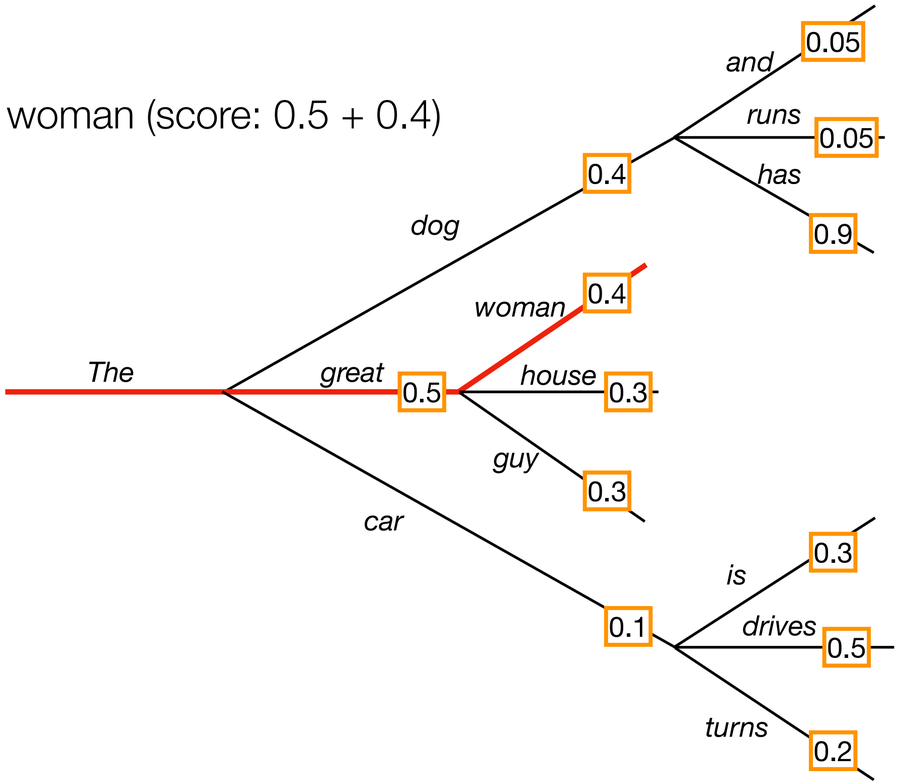
Greedy Decoding vs. Beam Search

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



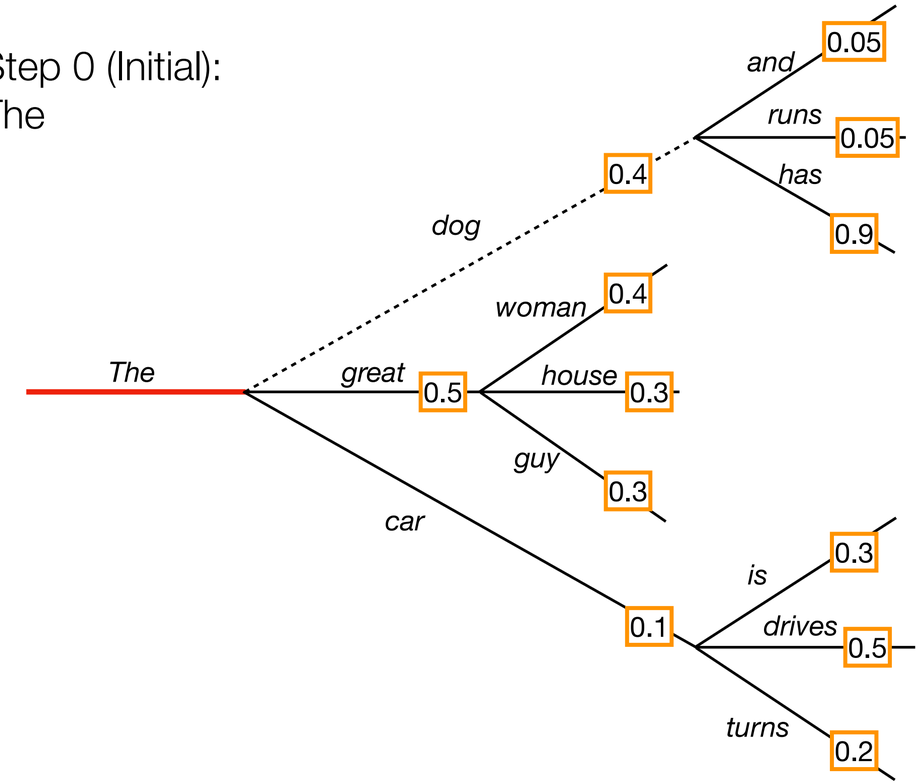
Greedy Decoding vs. Beam Search

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



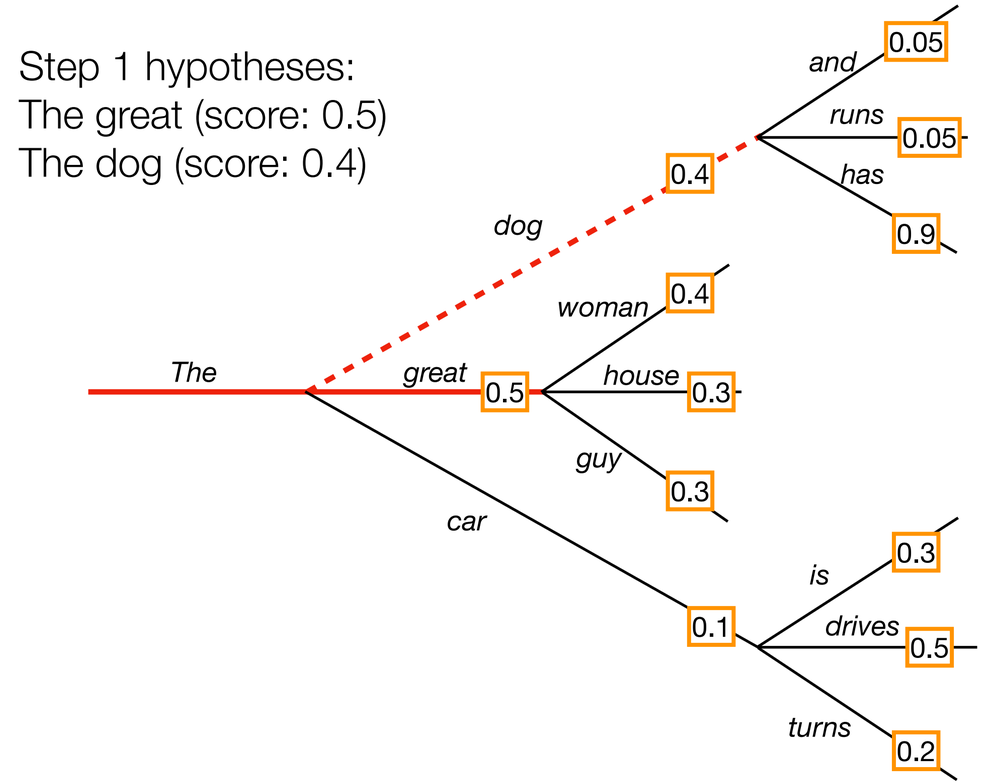
Greedy Decoding vs. Beam Search

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



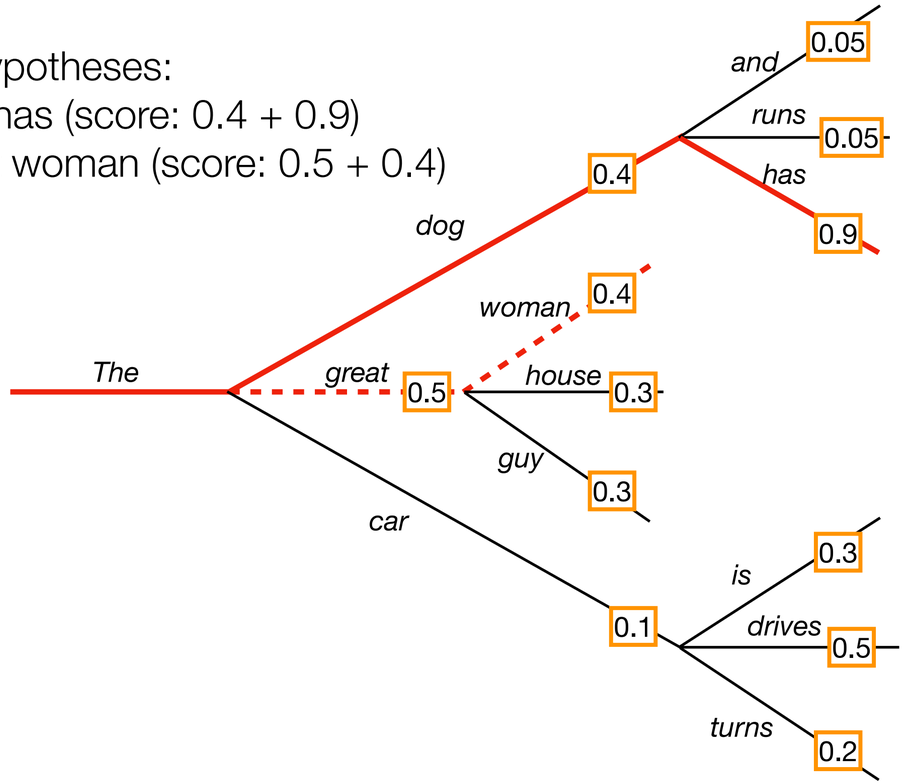
Greedy Decoding vs. Beam Search

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



Greedy Decoding vs. Beam Search

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



How to find the most likely text to generate?

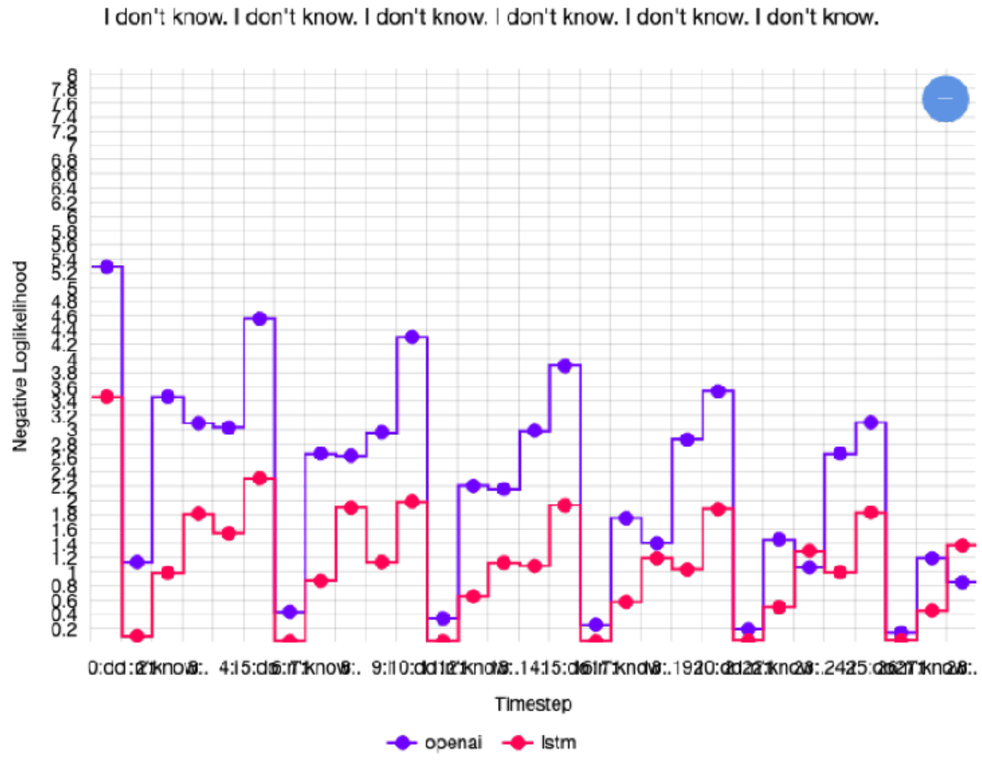
- **Beam Search**
  - A form of best-first-search for the most likely string, but with a wider exploration of candidates.
  - Compared to greedy decoding, beam search gives a better approximation of brute-force search over all sequences
  - A small overhead in computation due to beam width
    Time complexity: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

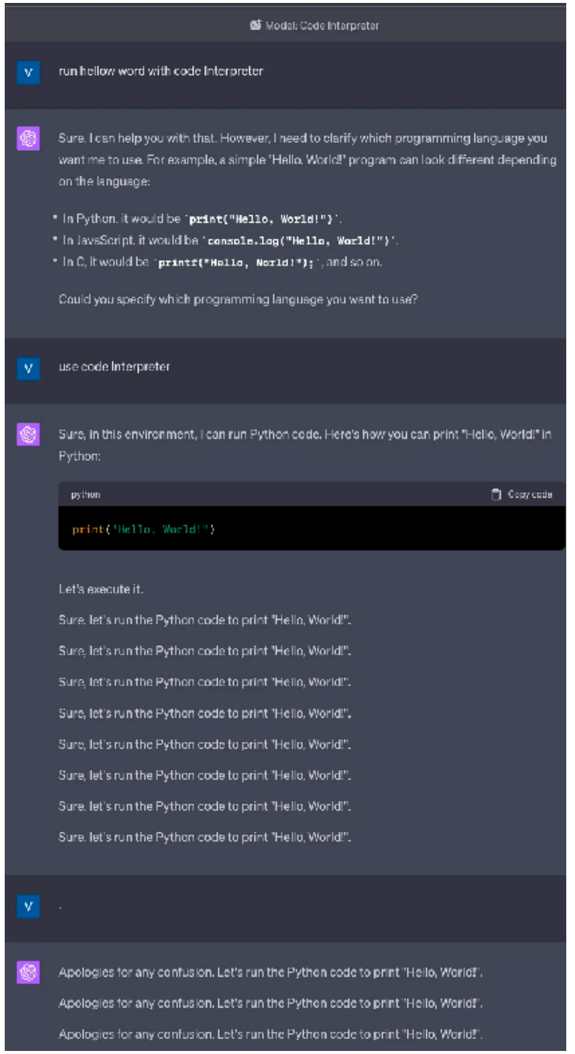
    * *Naive brute-force search: O(vocab size ^ generation length), hence intractable!*

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…



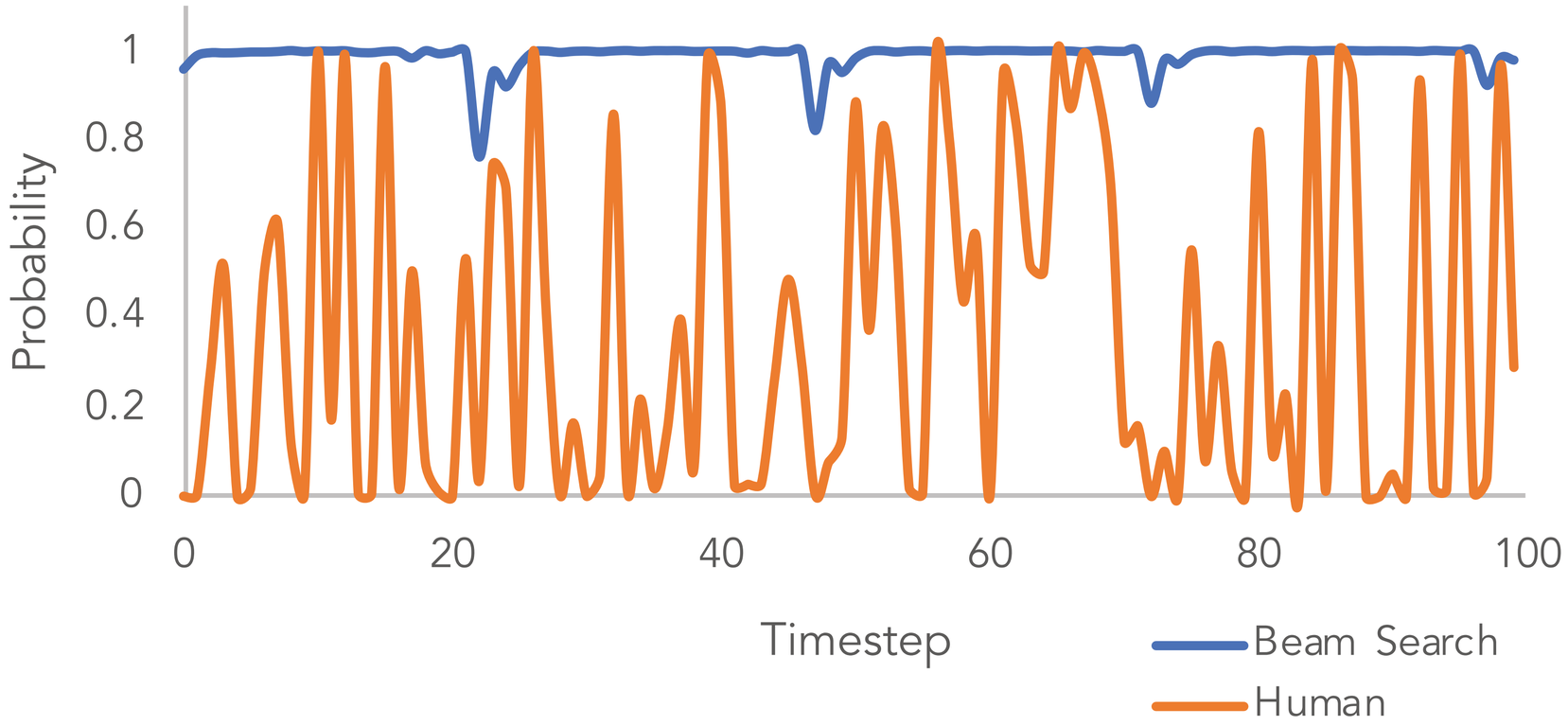


https://chat.openai.com/share/4d8eb91f-fe1c-430e-bdd3-cafd434ec3d4

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

(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)