Natural Language Generation I

Kirill Milintsevich | 11.01.2024

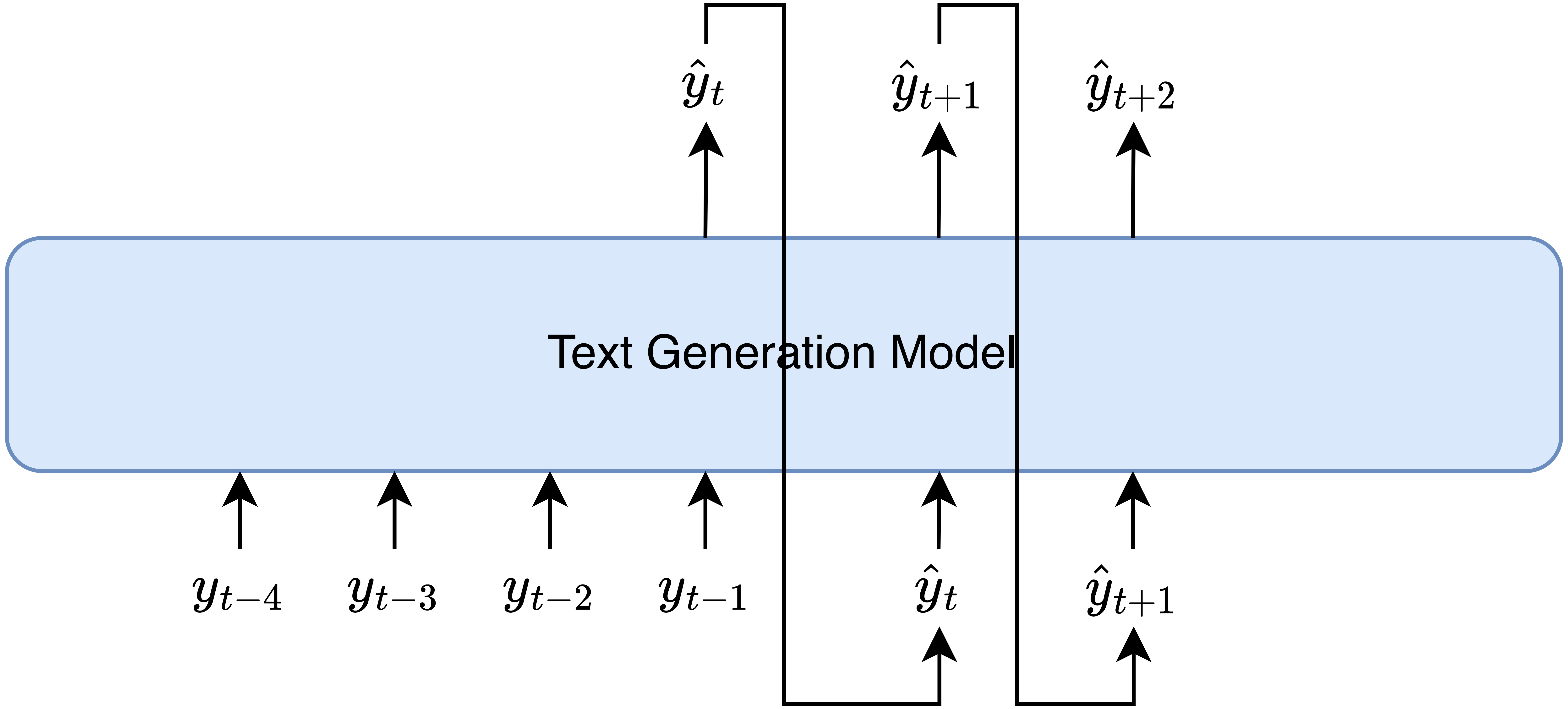
NLG Tasks

- Machine Translation
- Dialogue Systems
- Summarisation
- Data-to-text Generation
- Visual Description

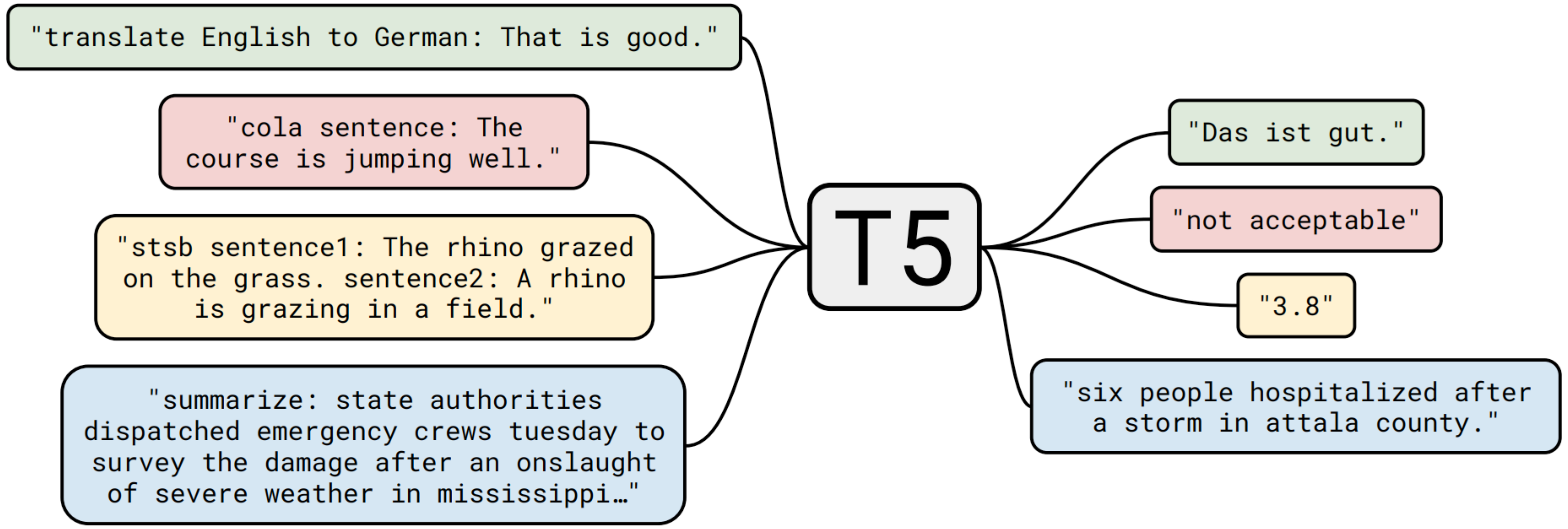
Autoregressive Models

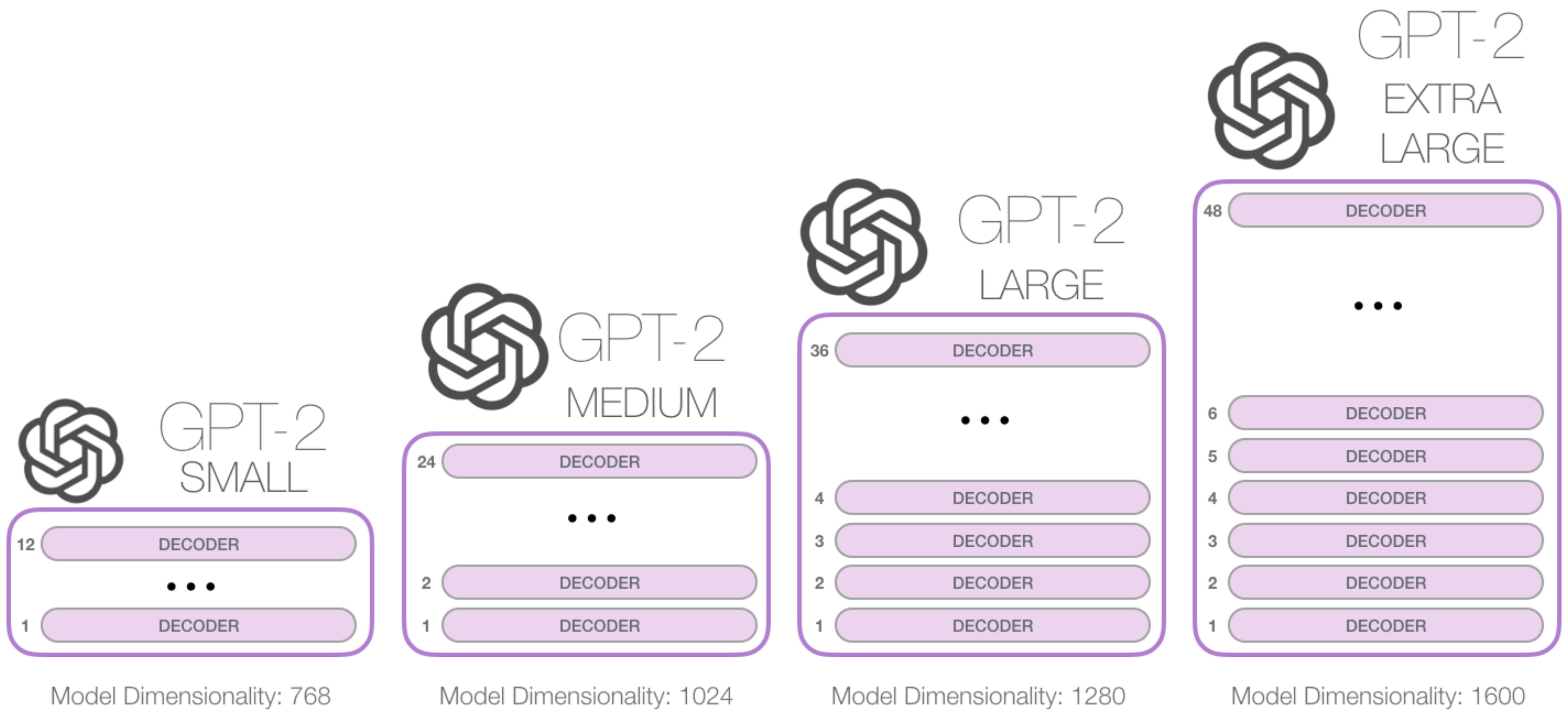
Autoregressive Models

- In autoregressive text generation models, at each time step t, our model takes in a sequence of tokens of text as input $y_{<t}$ and outputs a new token, $\hat{y}_t$

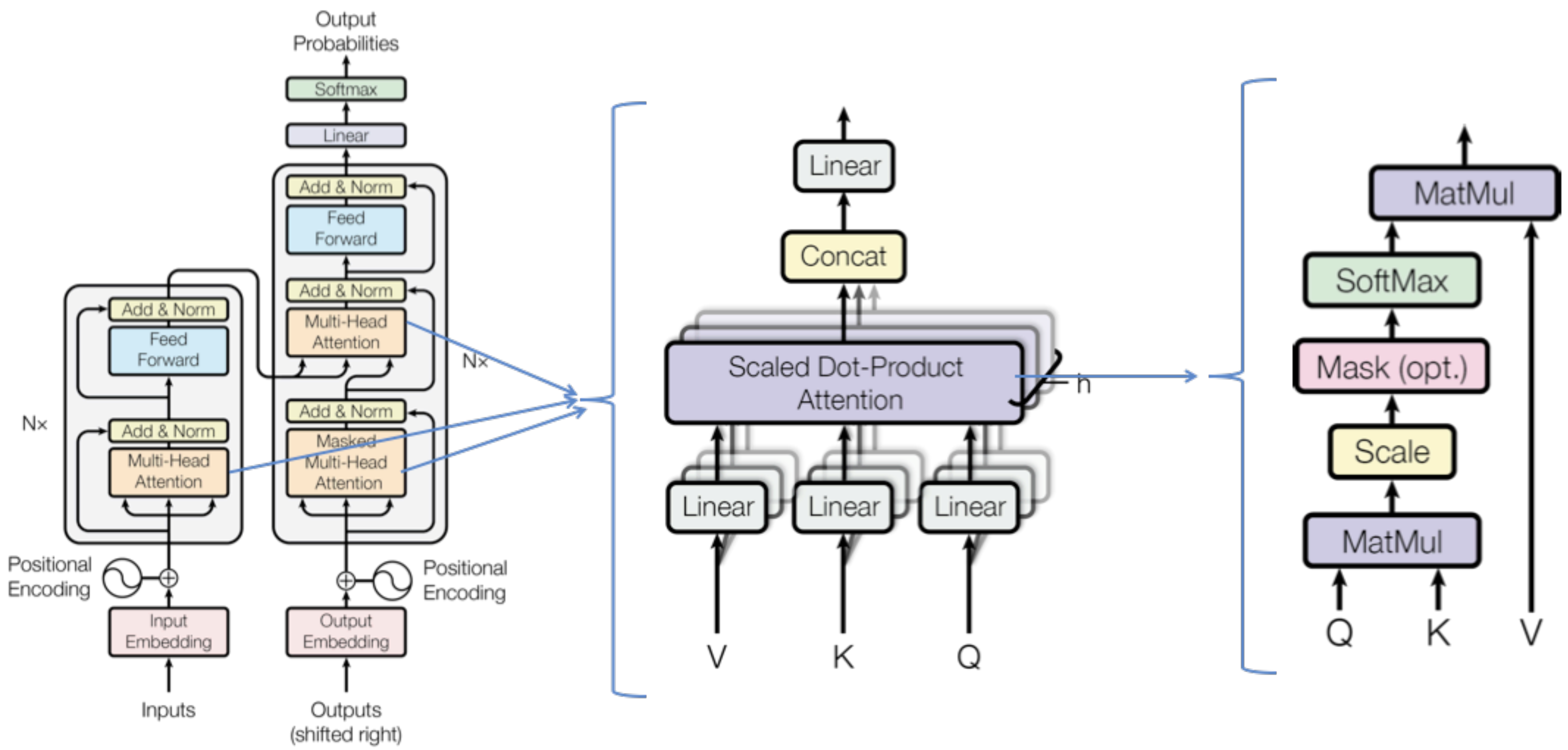


SOTA Autoregressive Models





Inside These Models



During a Single Step

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

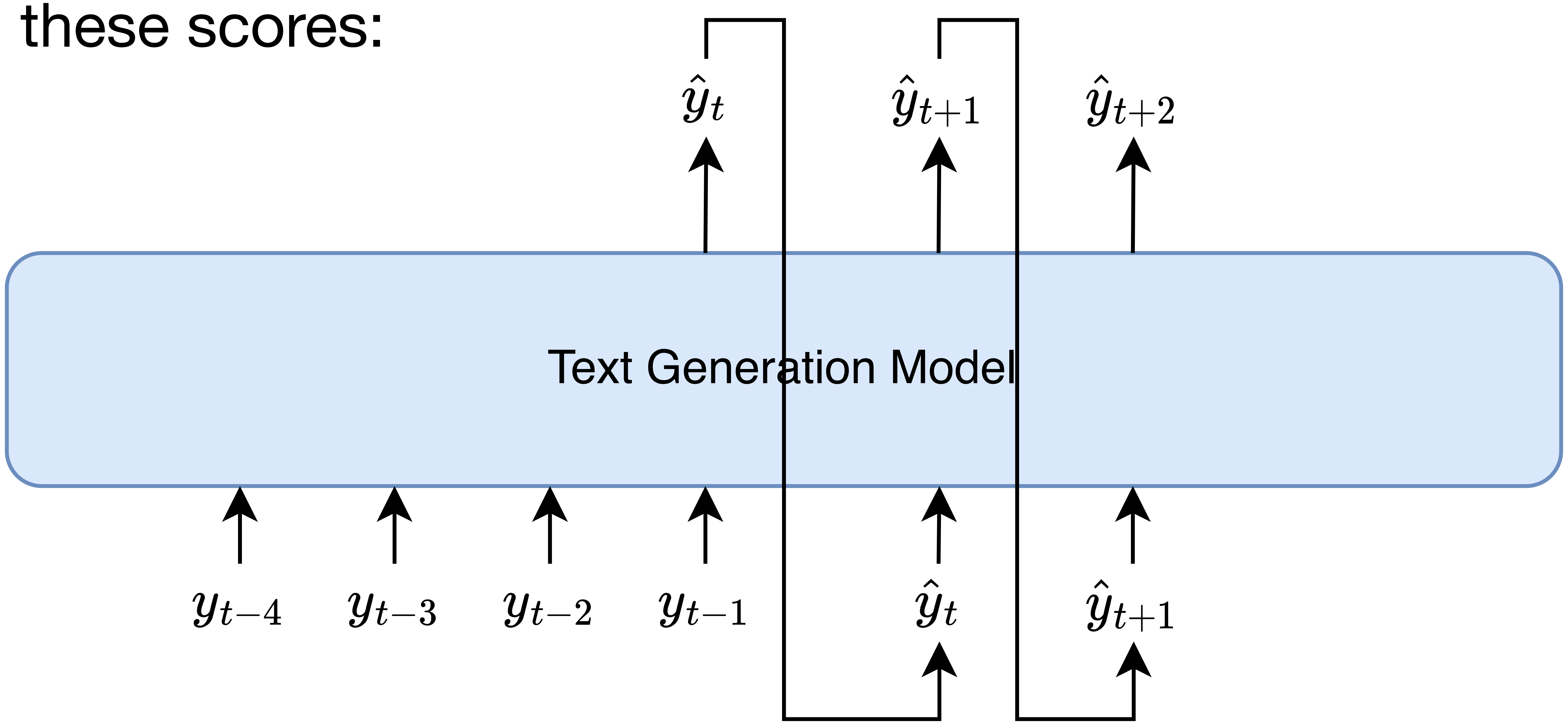
$$S = f(\{y_{<t}\}, \theta)$$

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

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

Autoregressive Models

- 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:



Inference and Training

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

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

- We train the model to minimize the negative loglikelihood of predicting the next token in the sequence:

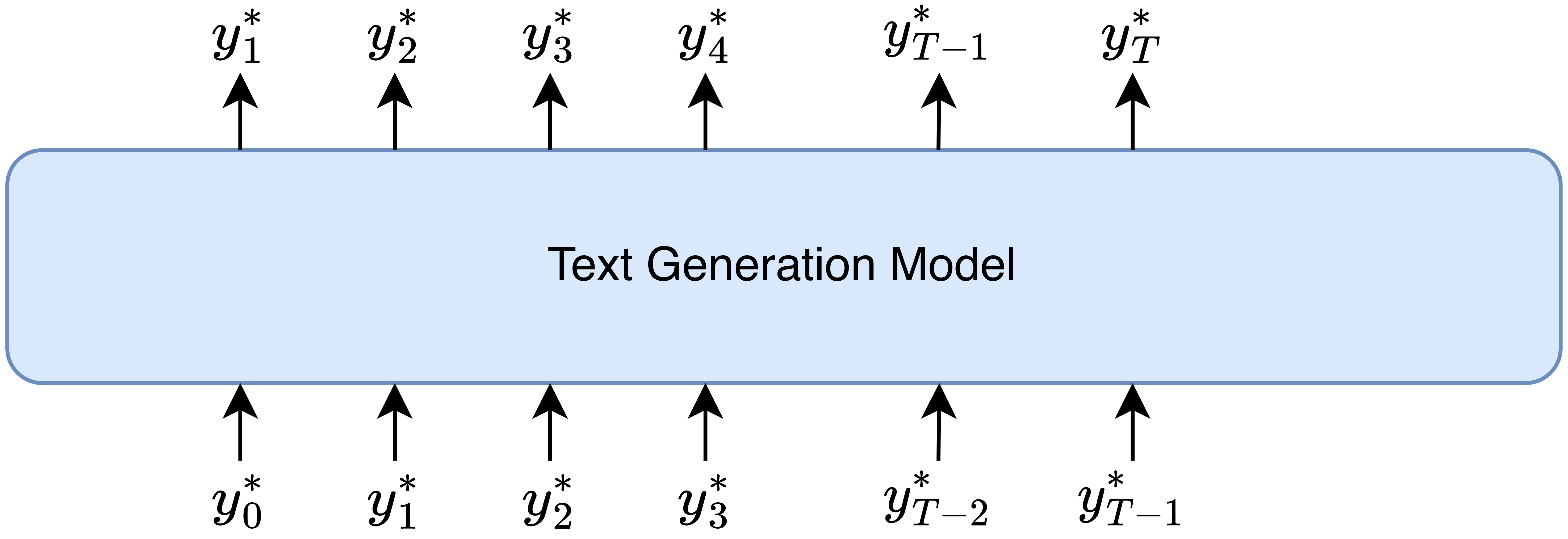
$$L_t = -\log P(y_t^* \quad \{y_{<t}^*\})$$

Maximum Likelihood Training

Teacher Forcing

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

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



Decoding

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

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

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

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

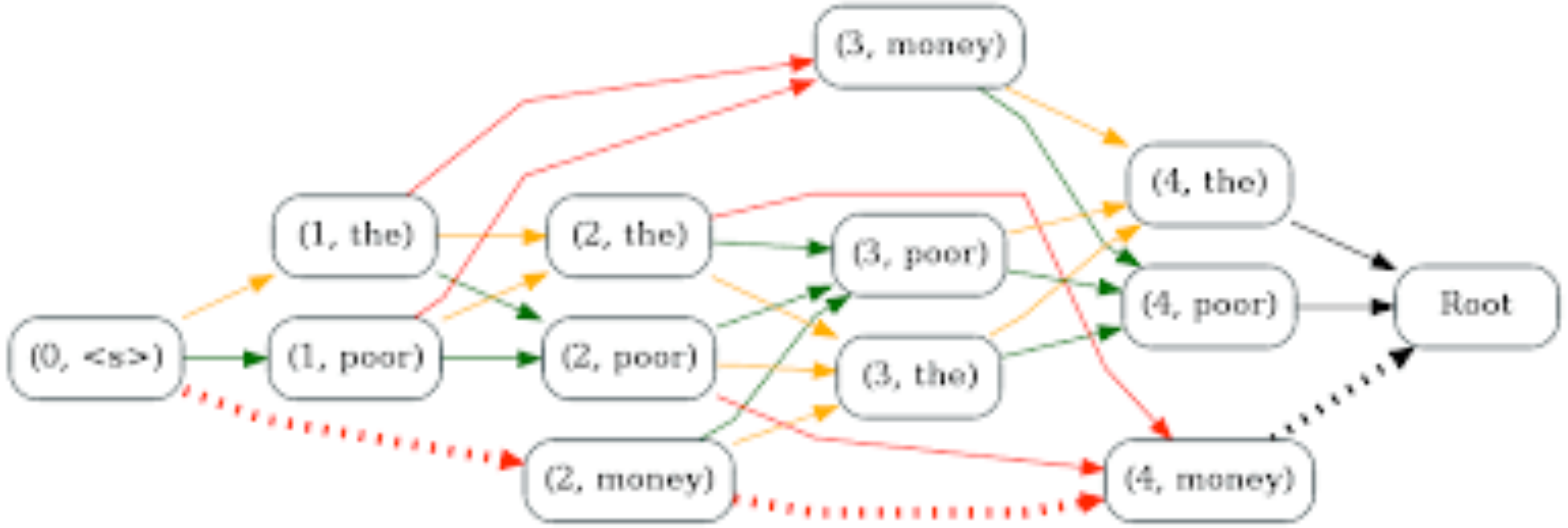
- Our decoding algorithm defines a function to select a token from this distribution:

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

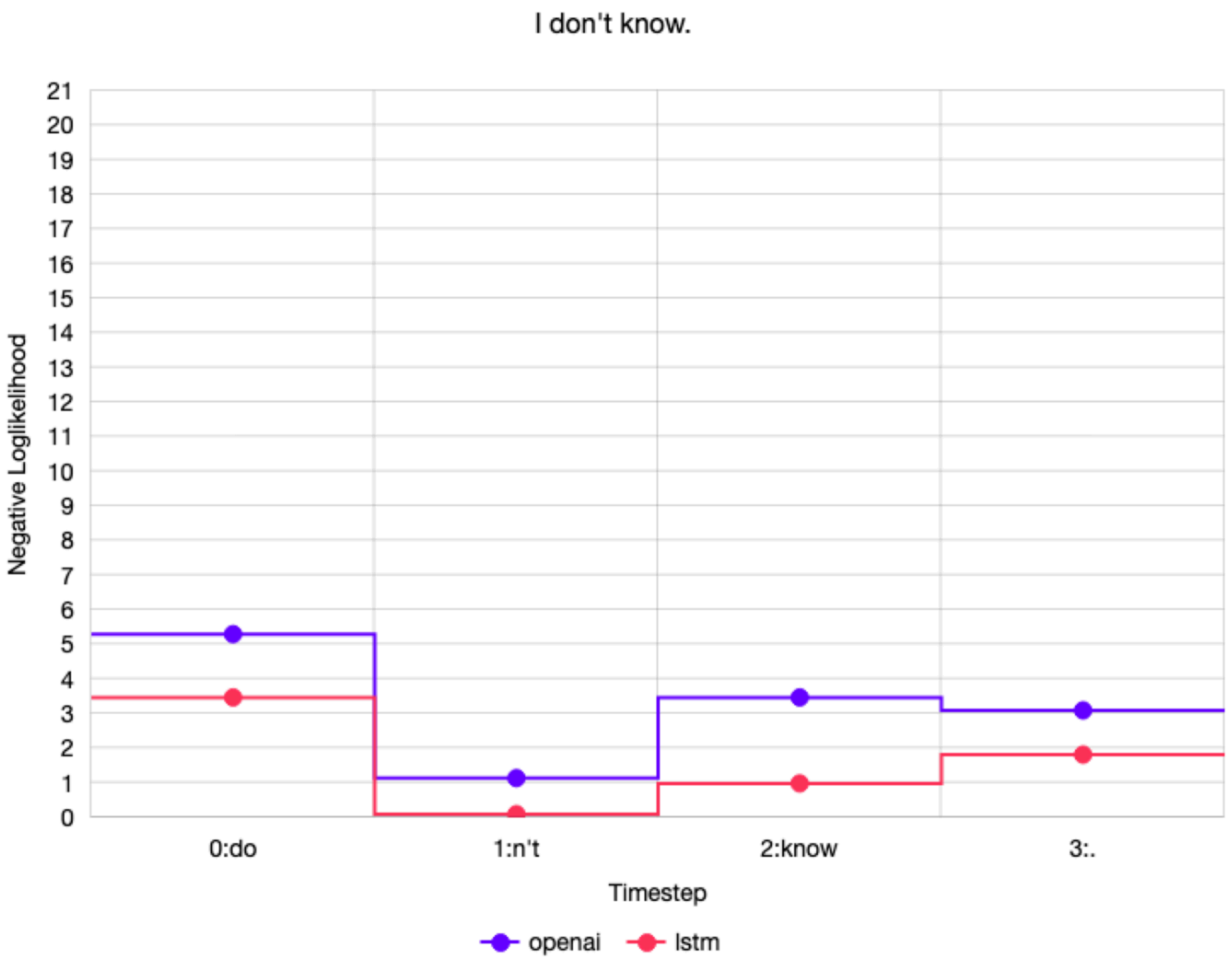
Greedy Methods

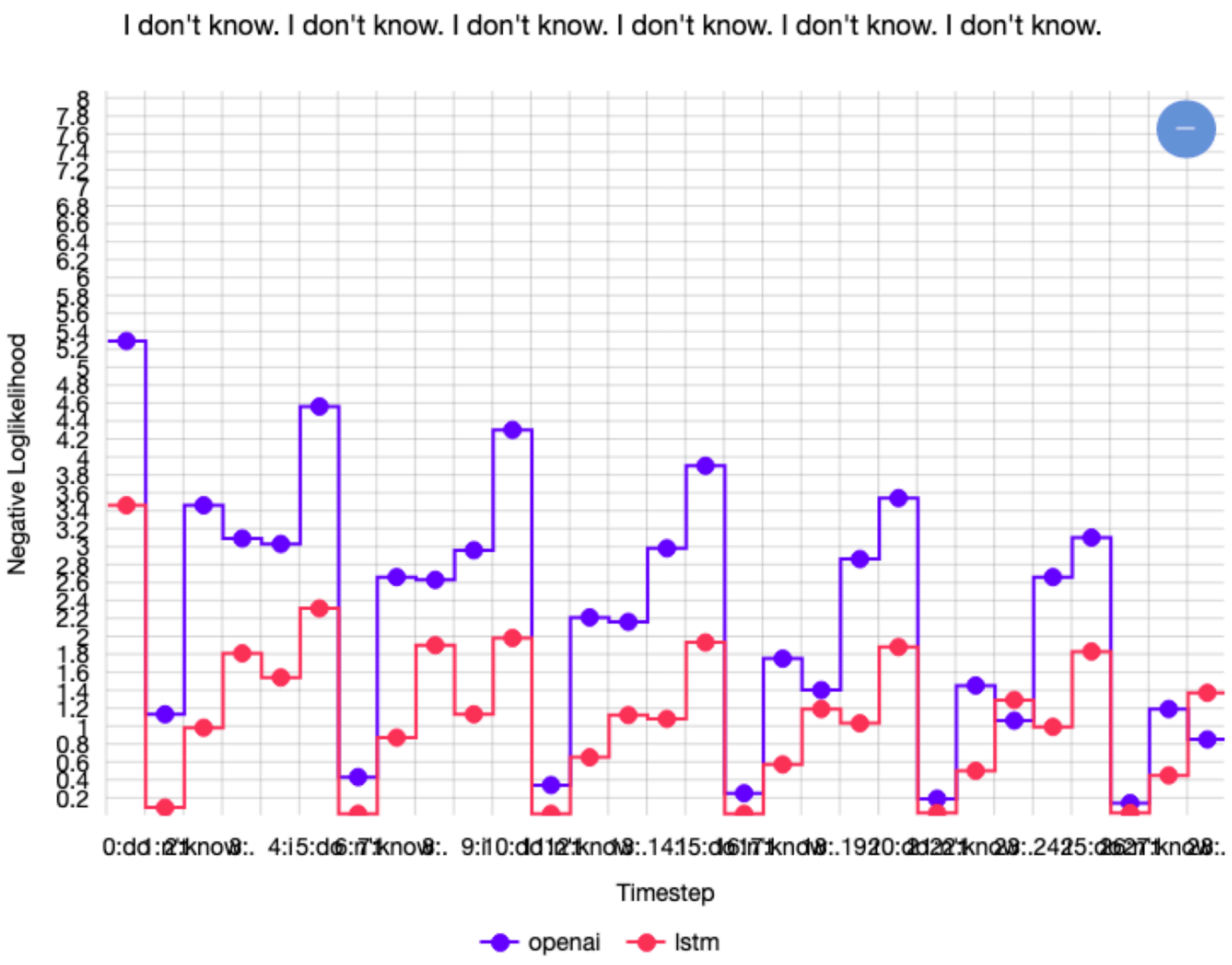
- Argmax Decoding
  - Selects the highest probability token in $P(y_t \;\; y_{<t})$

- Beam Search



Repetition in Greedy Methods





Sampling

- Sample a token from the distribution of tokens

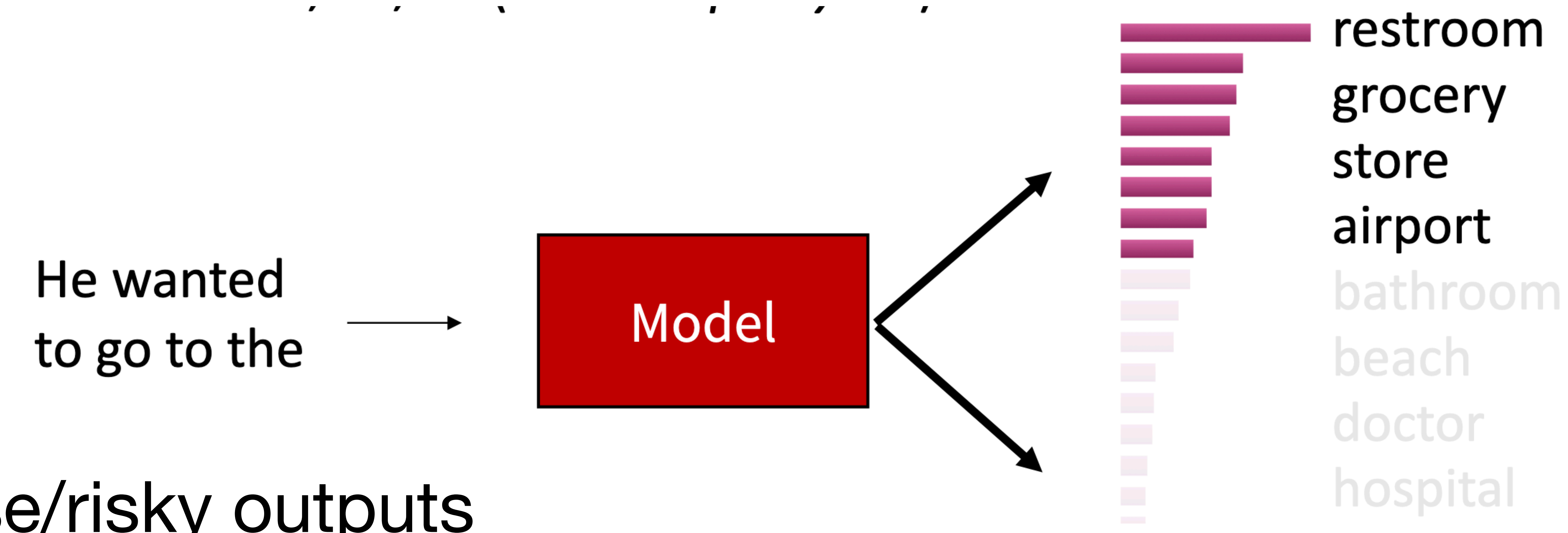
$$\hat{y}_t \sim P(y_t = w \mid \{y\}_{<t})$$

Top-k sampling

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

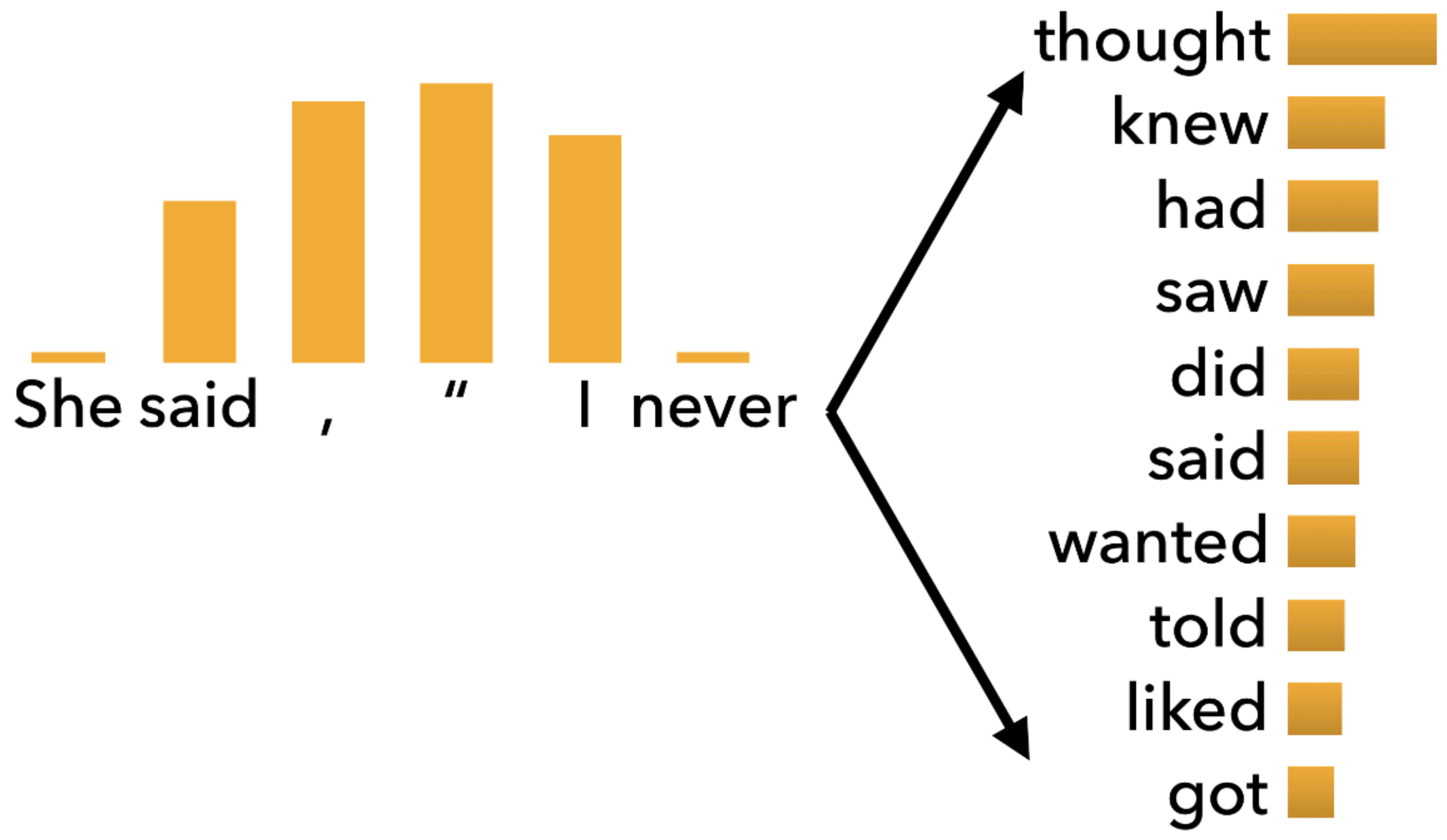
Top-k sampling

- Only sample from the top k tokens in the probability distribution
- Common values are k = 5, 10, 20 (but it's up to you!)

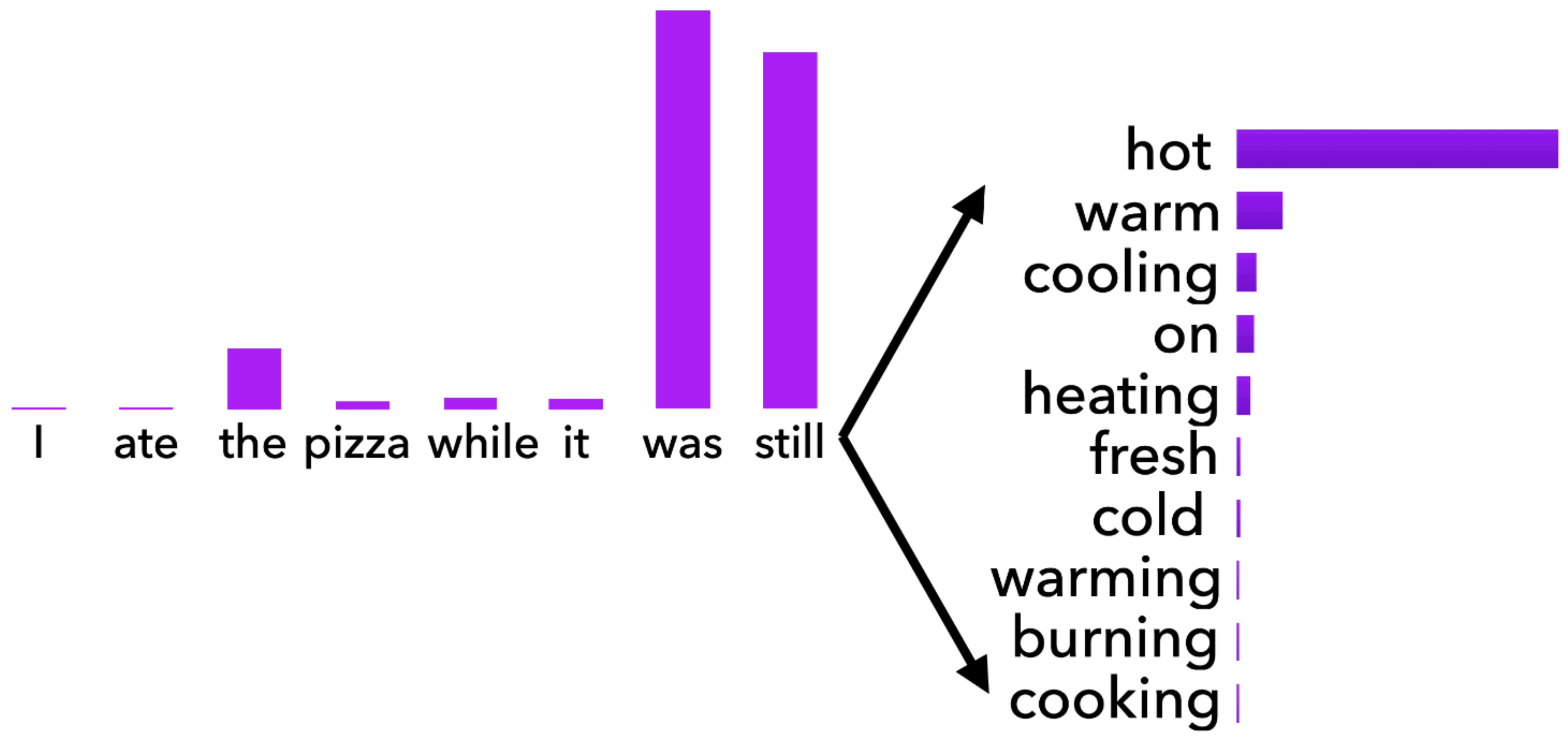


- Increase k for more diverse/risky outputs
- Decrease k for more generic/safe outputs

Top-k sampling



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



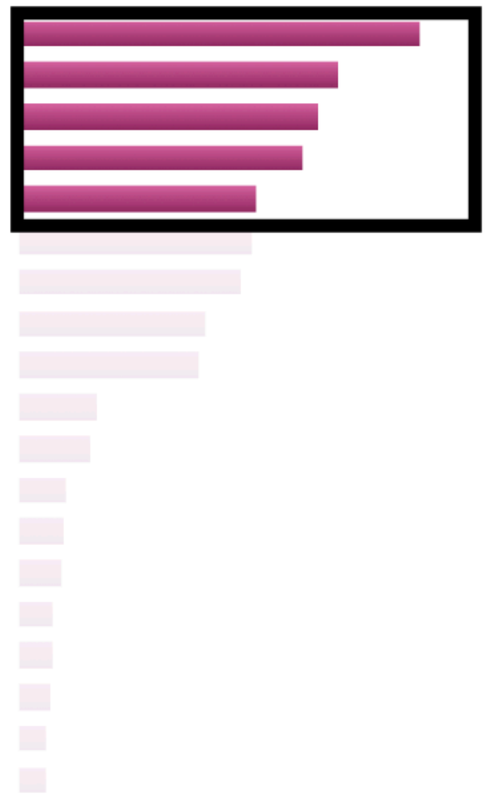
Top-*k* sampling can also cut off too ***slowly***!

Top-p (nucleus) sampling

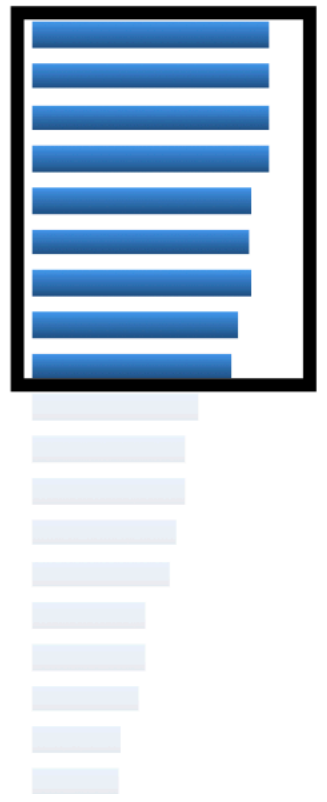
- Problem: The probability distributions we sample from are dynamic
  - When the distribution P_t is flatter, a limited k removes many viable options
  - When the distribution P_t is peakier, a high k allows for too many options to have a chance of being selected
- Solution: Top-p sampling
  - Sample from all tokens in the top p cumulative probability mass (i.e., where mass is concentrated)
  - Varies k depending on the uniformity of P_t

Top-p (nucleus) sampling

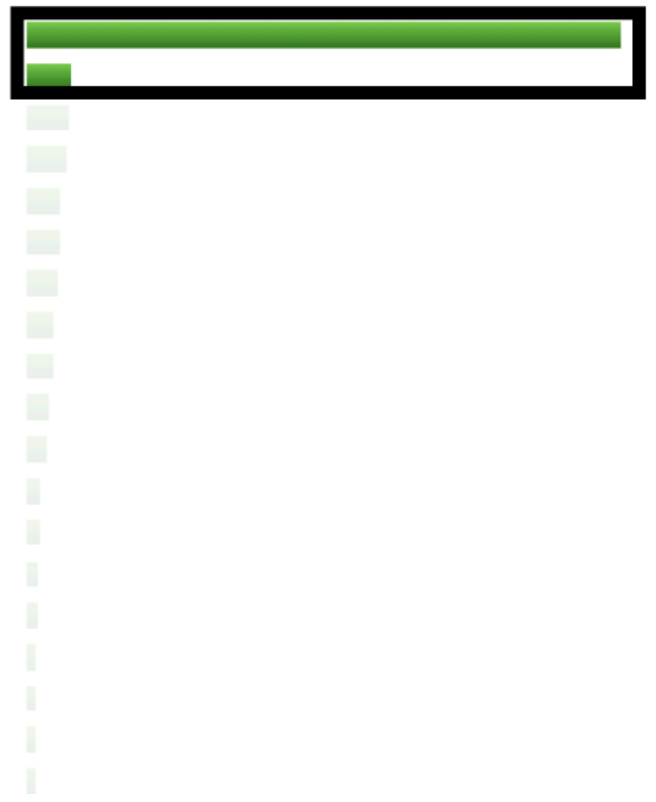
$P_t^1(y_t = w \mid \{y\}_{<t})$



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



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



Softmax Temperature

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

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

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

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

Softmax Temperature

- 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)

Re-ranking

- Decode a bunch of sequences
  - 10 candidates is a common number
- Define a score to approximate quality of sequences and re-rank by this score
  - Simplest is to use perplexity
    - Careful! Remember that repetitive methods can generally get high perplexity.

Decoding

- Decoding is still a challenging problem in natural language generation
- Human language distribution is noisy and doesn't reflect simple properties (i.e., probability maximization)
- 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

Evaluation

Content Overlap Metrics

Ref: They walked to the grocery store .

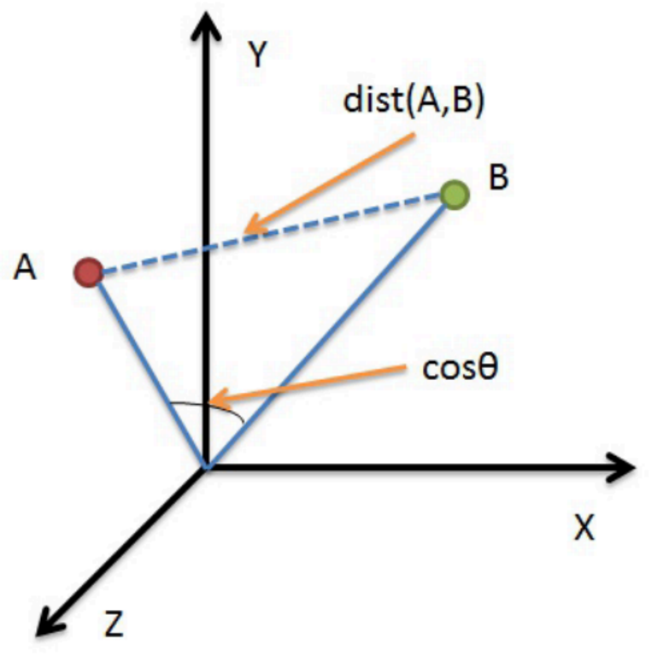
Gen: The woman went to the hardware store .

- Compute a score that indicates the similarity between generated and gold-standard (human-written) text
- Fast and efficient and widely used
- Two broad categories:
  - N-gram overlap metrics (e.g., BLEU, ROUGE, METEOR, CIDEr, etc.)
  - Semantic overlap metrics (e.g., PYRAMID, SPICE, SPIDEr, etc.)

Content Overlap Metrics

- They’re not ideal for machine translation
- They get progressively much worse for tasks that are more open-ended than machine translation
  - Worse for summarization, where extractive methods that copy from documents are preferred
  - Much worse for dialogue, which is more open-ended that summarization
  - Much, much worse story generation, which is also open-ended, but whose sequence length can make it seem you’re getting decent scores!

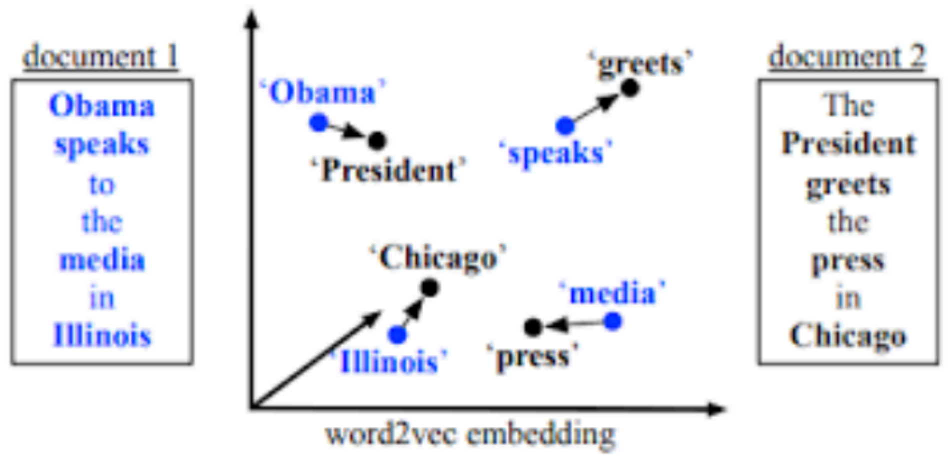
Model-Based Metrics



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)**

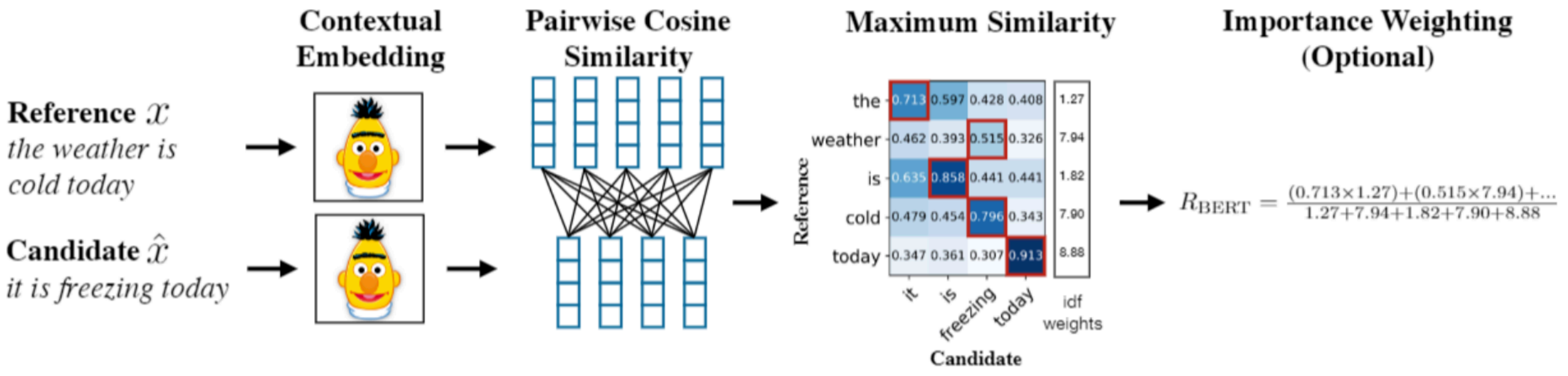


Word Mover's Distance:

Measures the distance between two sequences (e.g., sentences, paragraphs, etc.), 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. 2020)



Human Evaluation

- Ask humans to evaluate the quality of generated text
- Human judgments are regarded as the gold standard
- Humans are inconsistent

Evaluation

- Content overlap metrics provide a good starting point for evaluating the quality of generated text, but they’re not good enough on their own.
- Model-based metrics are can be more correlated with human judgment, but behavior is not interpretable
- Human judgments are critical.
  - Only ones that can directly evaluate factuality
  - But humans are inconsistent!

Non-autoregressive models

Non-autoregressive Models

Application	Example Source (S) and target (T) text	Use seq2seq
Machine translation	S: Turing studied at King's College, where he was awarded first-class honours in mathematics. T: Turing studierte am King's College, wo er erstklassige Auszeichnungen in Mathematik erhielt.	✓
Summarization	S: Court members Deborah Poritz and Peter Verniero did not participate in the Nelson case. T: Court members didn't participate in the case.	?
Sentence fusion	S: Turing was born in 1912. Turing died in 1954. T: Turing was born in 1912 and he died in 1954.	✗
Grammar correction	S: New Zealand have a cool weather. T: New Zealand has cool weather.	✗

Non-autoregressive Models

- Most NLP tasks apart from Machine Translation are monolingual
- Sources and targets often overlap
  - Generating the target from scratch is wasteful
  - Can reconstruct most of the target from the source via basic operations like KEEP, DELETE, INSERT

Turing	was	born	in	1912	.	Turing	died	in	1954	.
KEEP	KEEP	KEEP	KEEP	KEEP	DEL INS	PRON	KEEP	KEEP	KEEP	KEEP
Turing	was	born	in	1912	and	he	died	in	1954	.

Applications

- Grammatical Error Correction
- Text Simplification
- Sentence fusion
- Style transfer
- Text normalisation
- Text summarisation
- Automatic post-editing for machine translation

Advantages

- Text editing models need less training data
- They are faster at the inference
- They are more faithful
- We can control what model adds or removes
- We can incorporate external knowledge

Conclusion

Conclusion

- Natural Language Generation made a huge progress in the recent years
- NLG tasks cover a vast field of NLP problems (technically, any NLP task can be converted into a text generation!)
- Evaluating NLG models is challenging
- Training a performant NLG model requires a lot of data