香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

COMP 5212
Machine Learning
Lecture 24

Language Models, Pretraining

Junxian He
May 8, 2024

Reminder: HW4 due this Sunday

HW4 only has multi-choice questions to review the contents of this semester, expected to be finished within 2 hours

Recap: Learning a Proxy Model

Policy is a function parameterized by θ: π_θ

$\tau = (s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T)$ is the trajectory from π_θ

τ is a random variable

In practice, we often use a learned value function here

Total payoff for the policy is:

$$\eta_\theta = \mathbb{E}_{\tau \sim p_\theta}[\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)]$$

p_θ contains the policy and the transition p(s' | s, a)

We want to optimize θ to maximize η

Recap: Value Function Approximation

Learn a function $f_\theta(s) : s \rightarrow V(s)$

Similar to supervised learning, $f_\theta(s)$ could be a neural network

1. Randomly sample n states $s^{(1)}, s^{(2)}, s^{(3)} \ldots$

Want to compose a training dataset, next is to estimate V(s) for these n points

2. For every state, repeat value iteration to approximate $V(s^{(i)})$

3. Now we have a supervised dataset to train $f_\theta(s)$

Recap: Policy Gradient

$$\eta_\theta = \mathbb{E}_{\tau \sim p_\theta}[\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)]$$

We define $f(\tau) = \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)$

$$\eta(\theta) = \mathrm{E}_{\tau \sim P_\theta}\left[f(\tau)\right]$$

Connection to VAE? Reparameterization trick?

Recap: Policy Gradient

$$
\begin{aligned}
\nabla_\theta \mathrm{E}_{\tau\sim P_\theta}\left[f(\tau)\right] &= \nabla_\theta \int P_\theta(\tau) f(\tau) d\tau \\
&= \int \nabla_\theta (P_\theta(\tau) f(\tau)) d\tau \qquad \text{(swap integration with gradient)} \\
&= \int (\nabla_\theta P_\theta(\tau)) f(\tau) d\tau \qquad \text{(becaue } f \text{ does not depend on } \theta\text{)} \\
&= \int P_\theta(\tau) (\nabla_\theta \log P_\theta(\tau)) f(\tau) d\tau
\end{aligned}
$$

$$
= \mathrm{E}_{\tau\sim P_\theta}\left[(\nabla_\theta \log P_\theta(\tau)) f(\tau)\right]
$$

Can be approximated using MC sampling

Policy gradient is a commonly used method to propagate gradients through discrete variables

Recap: Policy Gradient

$$\mathrm{E}_{\tau\sim P_\theta}\left[(\nabla_\theta \log P_\theta(\tau))f(\tau)\right]$$

What is $\nabla_\theta \log P_\theta(\tau)$

$$P_\theta(\tau) = \mu(s_0)\pi_\theta(a_0|s_0)P_{s_0a_0}(s_1)\pi_\theta(a_1|s_1)P_{s_1a_1}(s_2)\cdots P_{s_{T-1}a_{T-1}}(s_T)$$

$$\nabla_\theta \log P_\theta(\tau) = \nabla_\theta \log \pi_\theta(a_0|s_0) + \nabla_\theta \log \pi_\theta(a_1|s_1) + \cdots + \nabla_\theta \log \pi_\theta(a_{T-1}|s_{T-1})$$

$$\begin{aligned}\nabla_\theta \eta(\theta) = \nabla_\theta \mathrm{E}_{\tau\sim P_\theta}\left[f(\tau)\right] &= \mathrm{E}_{\tau\sim P_\theta}\left[\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\right)\cdot f(\tau)\right] \\ &= \mathrm{E}_{\tau\sim P_\theta}\left[\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\right)\cdot\left(\sum_{t=0}^{T-1}\gamma^t R(s_t,a_t)\right)\right]\end{aligned}$$

Loss does not mean much, and you should only care about the return

Learning Policy and Value Function Together

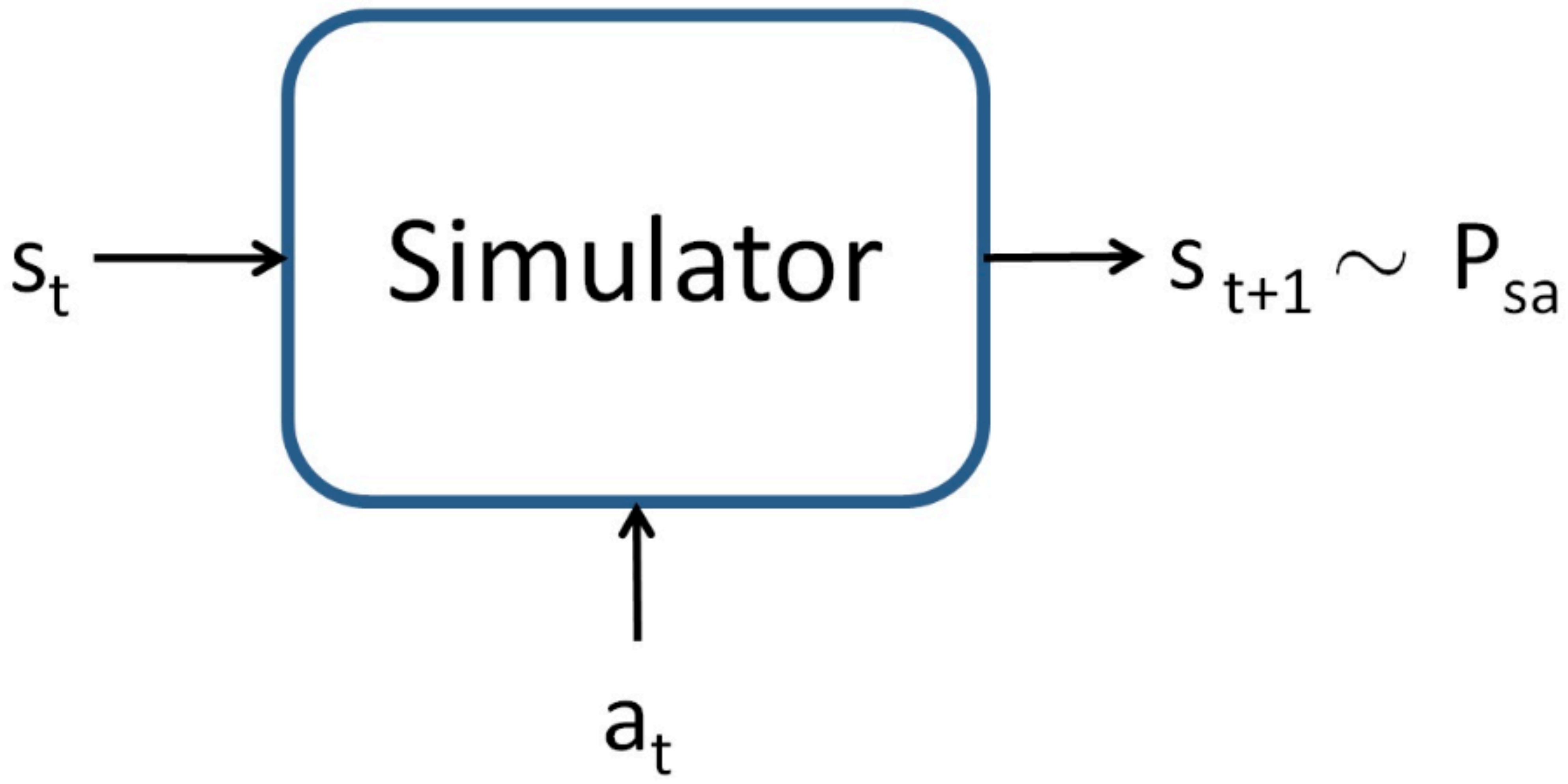
Repeat{

1. Perform a number of trials from policy π_θ to get all the trajectory
2. Update the policy with the current value function
3. Compute the expected reward for each state in the trajectories
4. Supervised training to train the value function

}

Reward models are often trained in advance

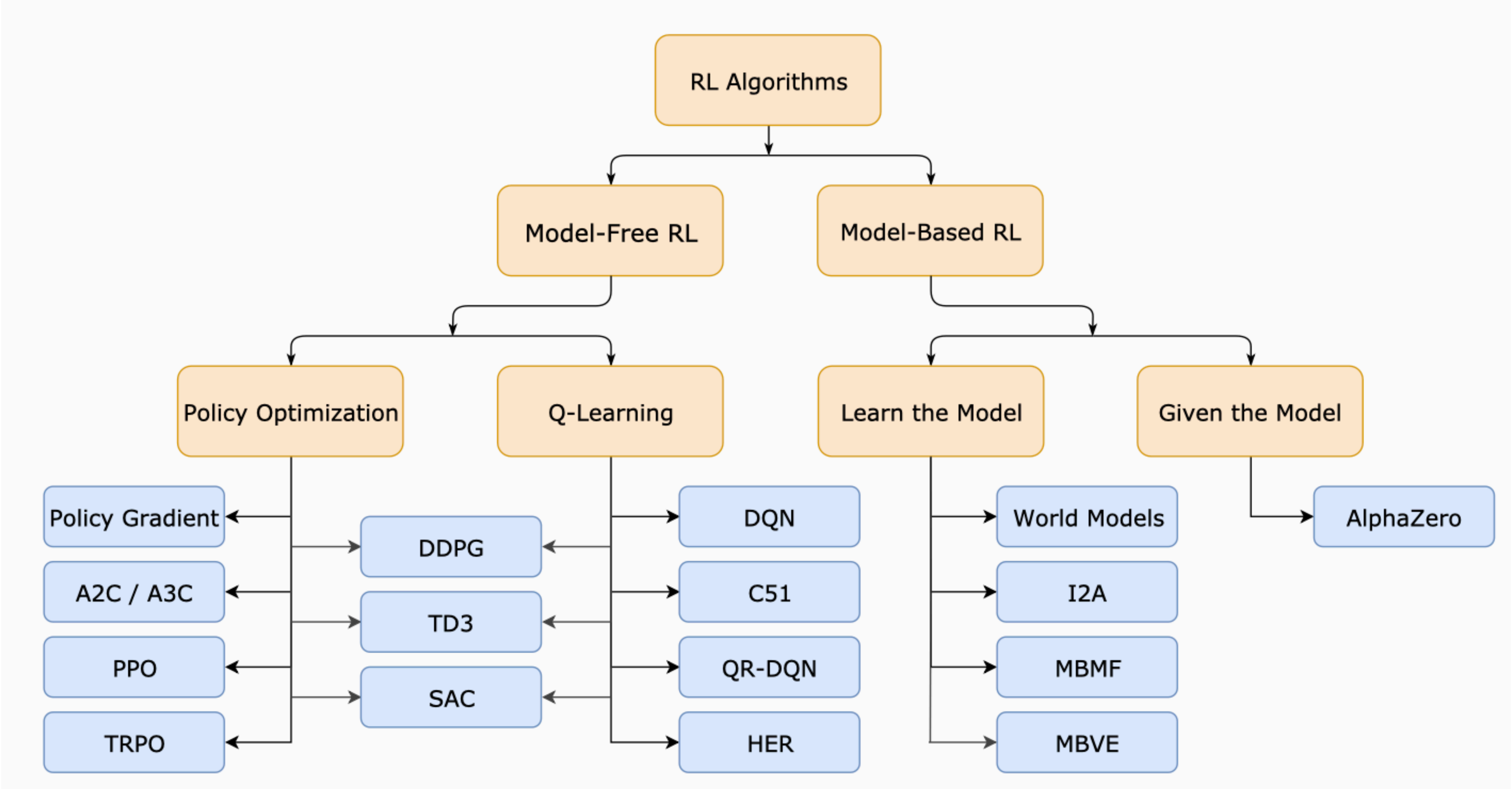
Model for the Environment



Model-based Reinforcement Learning

1. Interaction with real environment can be slow
2. Interaction with real environment can be risky

Taxonomy of RL



https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#part-2-kinds-of-rl-algorithms

Language Models

Probability of Sequences

Probability of multiple random variables:

$$p(x_1, x_2, \ldots, x_I) = \prod_{i=1}^{I} p(x_i \,|\, x_{1:i-1})$$

Probability of language:

$$\begin{aligned} p(\text{the, mouse, ate, the, cheese}) = \; & p(\text{the}) \\ & p(\text{mouse} \mid \text{the}) \\ & p(\text{ate} \mid \text{the, mouse}) \\ & p(\text{the} \mid \text{the, mouse, ate}) \\ & p(\text{cheese} \mid \text{the, mouse, ate, the}). \end{aligned}$$

Autoregressive language models

Autoregressive Language Models

$$\begin{aligned} p(\text{the, mouse, ate, the, cheese}) = p(&\text{the}) \\ p(&\text{mouse} \mid \text{the}) \\ p(&\text{ate} \mid \text{the, mouse}) \\ p(&\text{the} \mid \text{the, mouse, ate}) \\ p(&\text{cheese} \mid \text{the, mouse, ate, the}). \end{aligned}$$

$$p(x_1, x_2, \ldots, x_I) = \prod_{i=1}^{I} p(x_i \mid x_{1:i-1})$$

Next Word Context

Autoregressive Language Models

$$\begin{aligned} p(\text{the, mouse, ate, the, cheese}) = {} & p(\text{the}) \\ & p(\text{mouse} \mid \text{the}) \\ & p(\text{ate} \mid \text{the, mouse}) \\ & p(\text{the} \mid \text{the, mouse, ate}) \\ & p(\text{cheese} \mid \text{the, mouse, ate, the}). \end{aligned}$$

$$p(x_1, x_2, \ldots, x_I) = \prod_{i=1}^{I} p(x_i \mid x_{1:i-1})$$

Learning a language model is to learn these conditional probabilities, for any language sequence

Autoregressive Language Models

$$
\begin{aligned}
p(\text{the, mouse, ate, the, cheese}) = \; & p(\text{the}) \\
& p(\text{mouse} \mid \text{the}) \\
& p(\text{ate} \mid \text{the, mouse}) \\
& p(\text{the} \mid \text{the, mouse, ate}) \\
& p(\text{cheese} \mid \text{the, mouse, ate, the}).
\end{aligned}
$$

$$p(x_1, x_2, \ldots, x_I) = \prod_{i=1}^{I} p(x_i \,|\, x_{1:i-1})$$

Given a dataset, how to find these probabilities?

Maximum Likelihood Estimation

Count-based Language Models

Count the frequency and divide

$$p(x_i \mid x_{1:i-1}) = \frac{c(x_{1:i})}{c(x_{1:i-1})}$$

There are infinite number of parameters for language

We may see long sequences only once, counting becomes meaningless

n-gram Language Models

Next token probability only depends on the previous n-1 words

Unigram LM:

$$p(x_1, x_2, \ldots, x_I) = \prod_{i=1}^{I} p(x_i)$$

Each token is independent

Bigram LM:

$$p(x_1, x_2, \ldots, x_I) = \prod_{i=1}^{I} p(x_i \mid x_{i-1})$$

Markov assumption?

Generally for n-gram LM:

$$p(x_1, x_2, \ldots, x_I) = \prod_{i=1}^{I} p(x_i \mid x_{i-n+1:i-1})$$

Similar to n-th order HMM?
Is HMM autoregressive LM?

Parameter Estimation for n-gram LM

Count-based:

$$p(x_i \mid x_{i-n+1:i-1}) = \frac{c(x_{i-n+1:i})}{c(x_{i-n+1:i-1})}$$

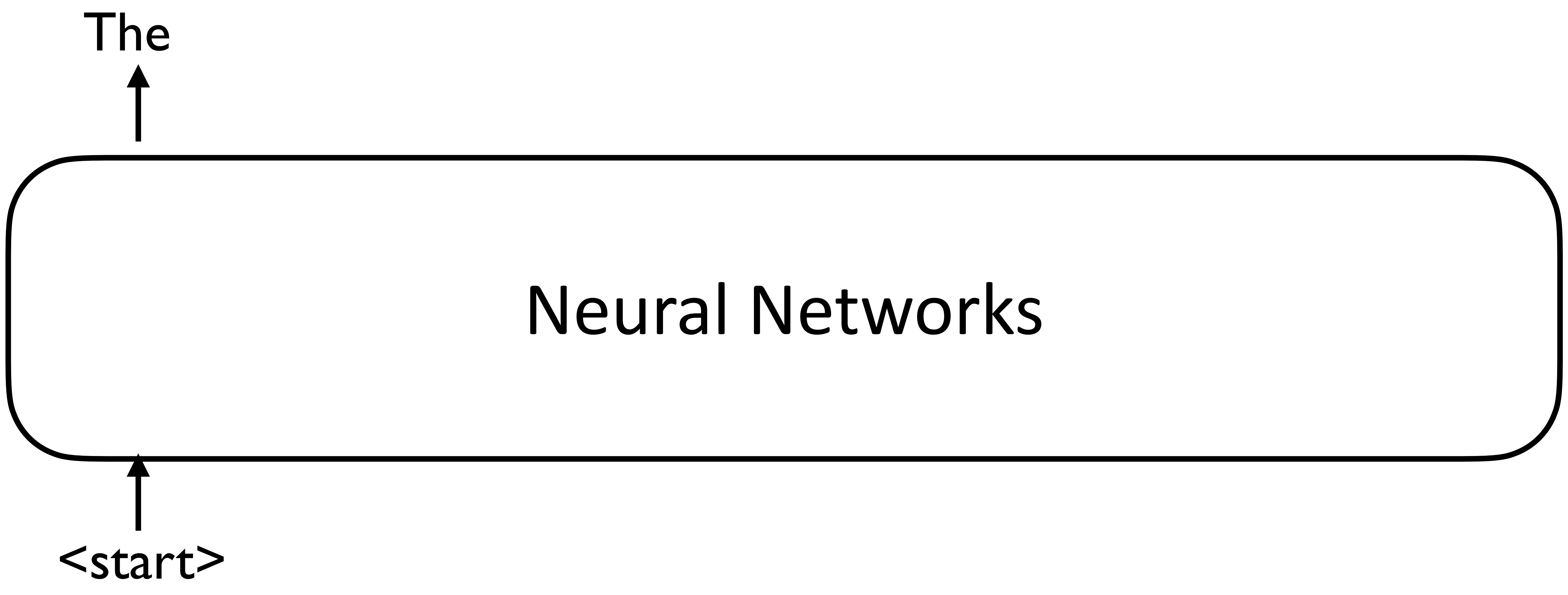
Number of parameters decreases, but flexibility decreases as well

Traditionally, we directly compute this probability, but neural language models use neural networks to compute the probability

Neural Language Models

Neural language models are typically autoregressive

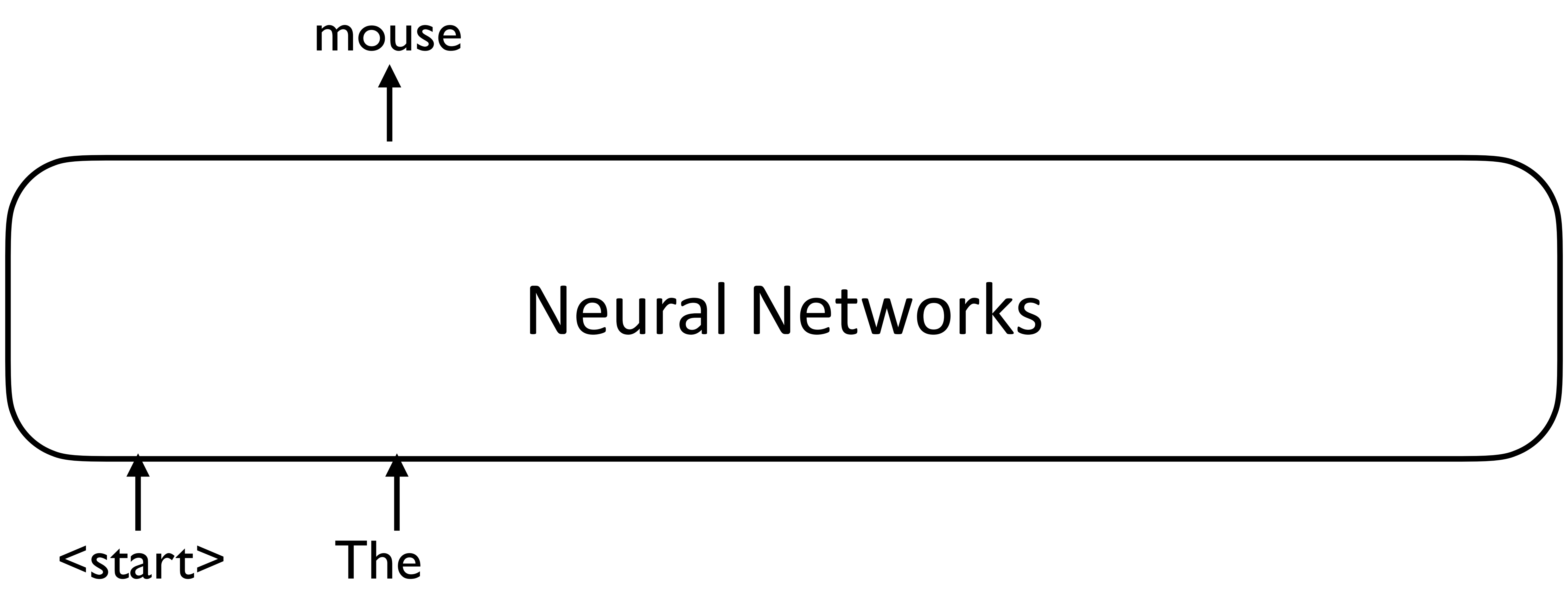
Data: “The mouse ate the cheese .”



Neural Language Models

Neural language models are typically autoregressive

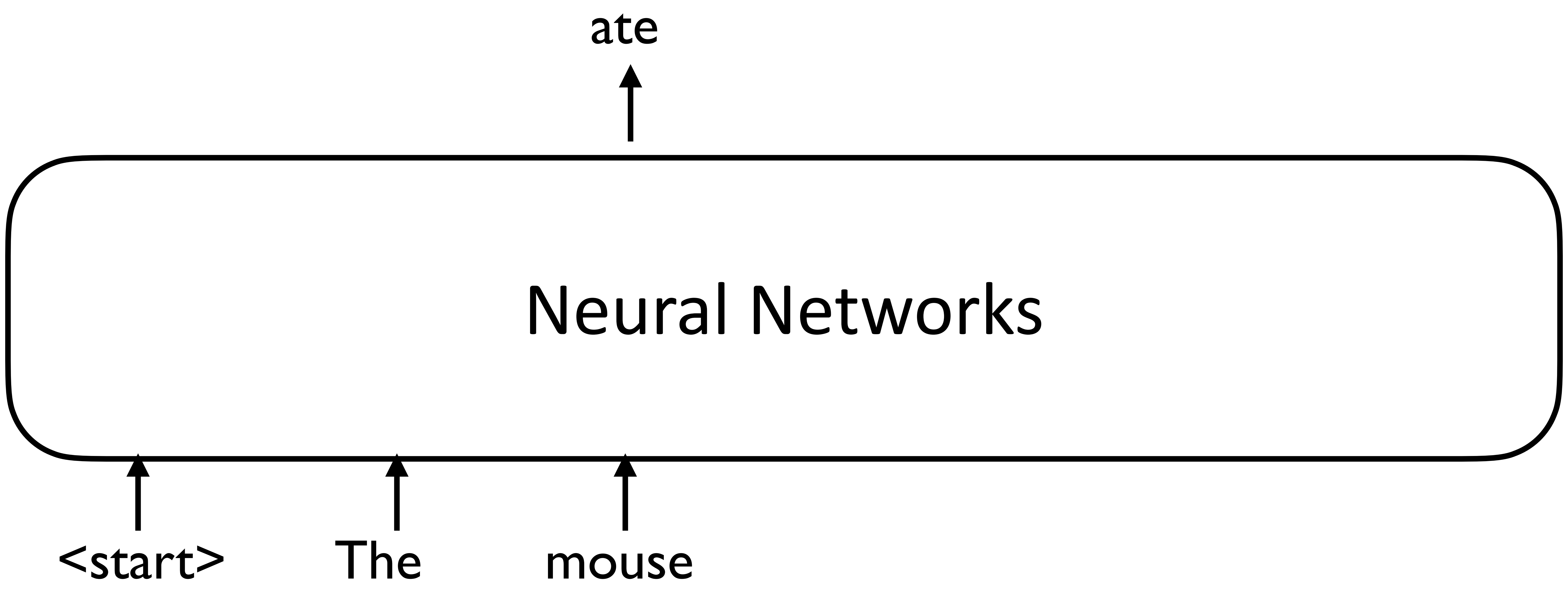
Data: "The mouse ate the cheese ."



Neural Language Models

Neural language models are typically autoregressive

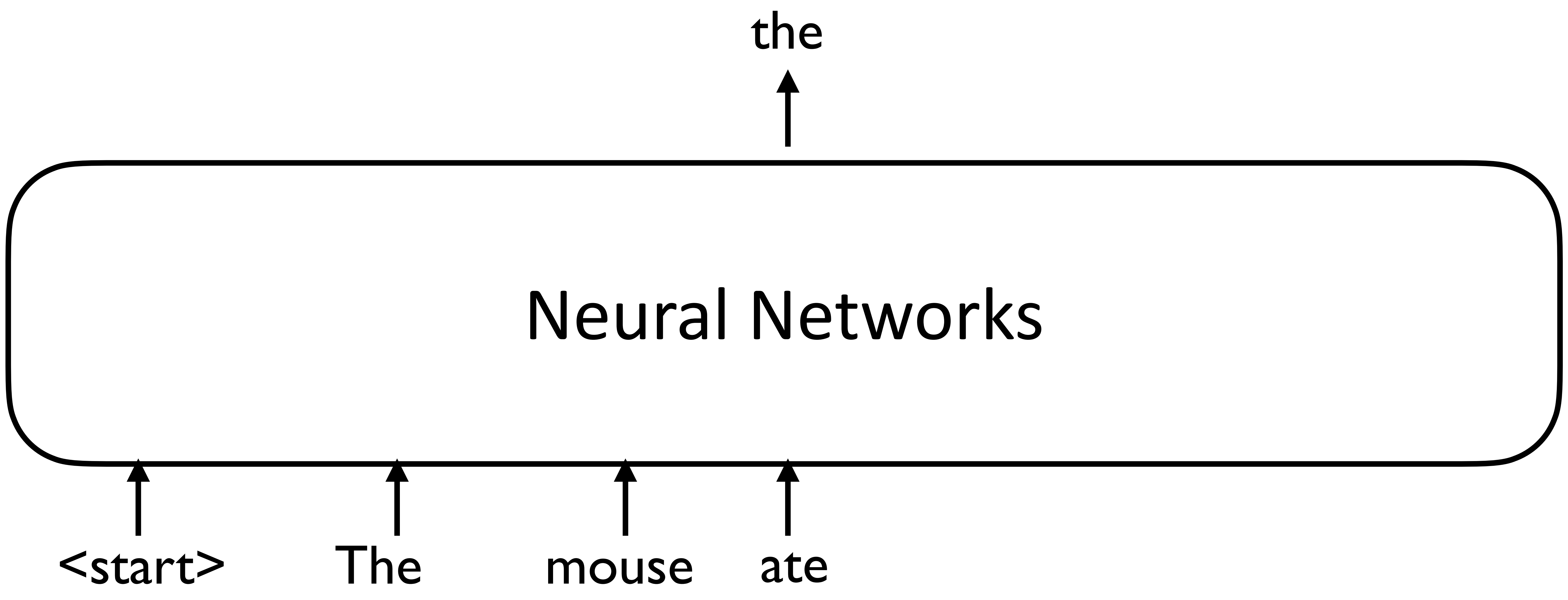
Data: "The mouse ate the cheese ."



Neural Language Models

Neural language models are typically autoregressive

Data: "The mouse ate the cheese ."

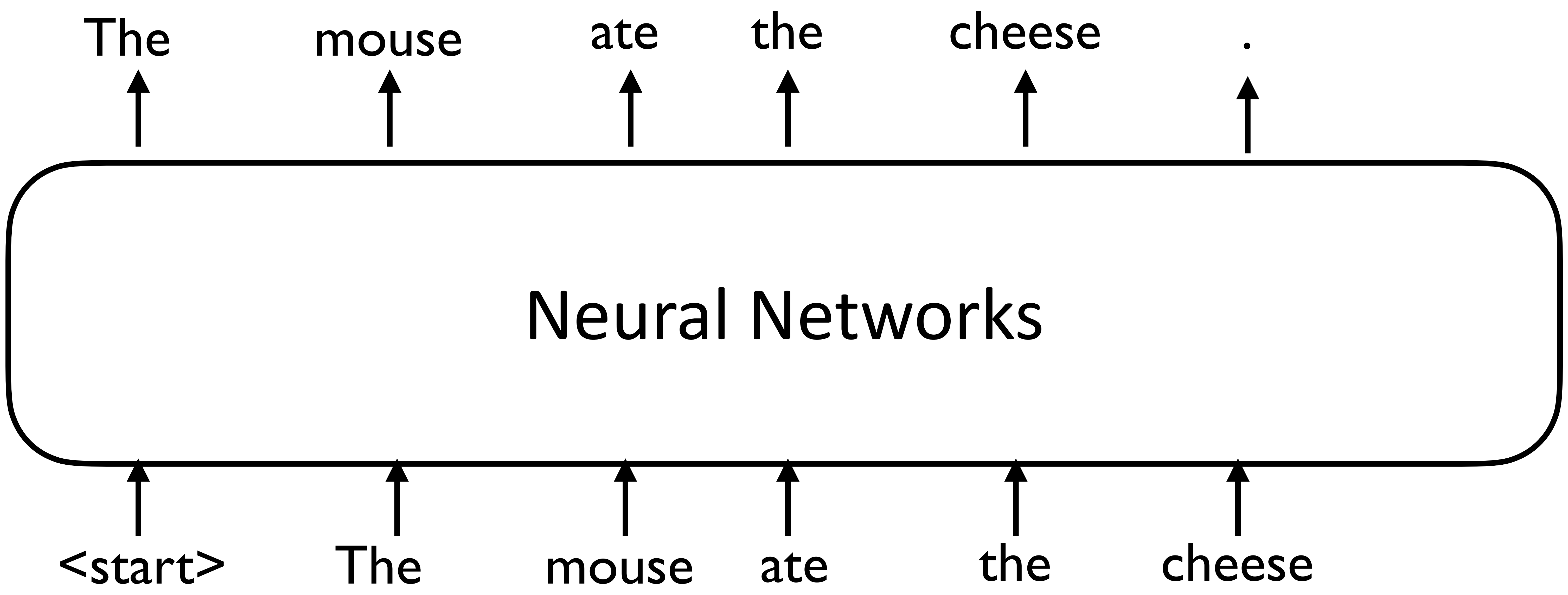


We can compute the loss on every token in parallel

Neural Language Models

Neural language models are typically autoregressive

Data: "The mouse ate the cheese ."



Each prediction only sees the inputs on its left

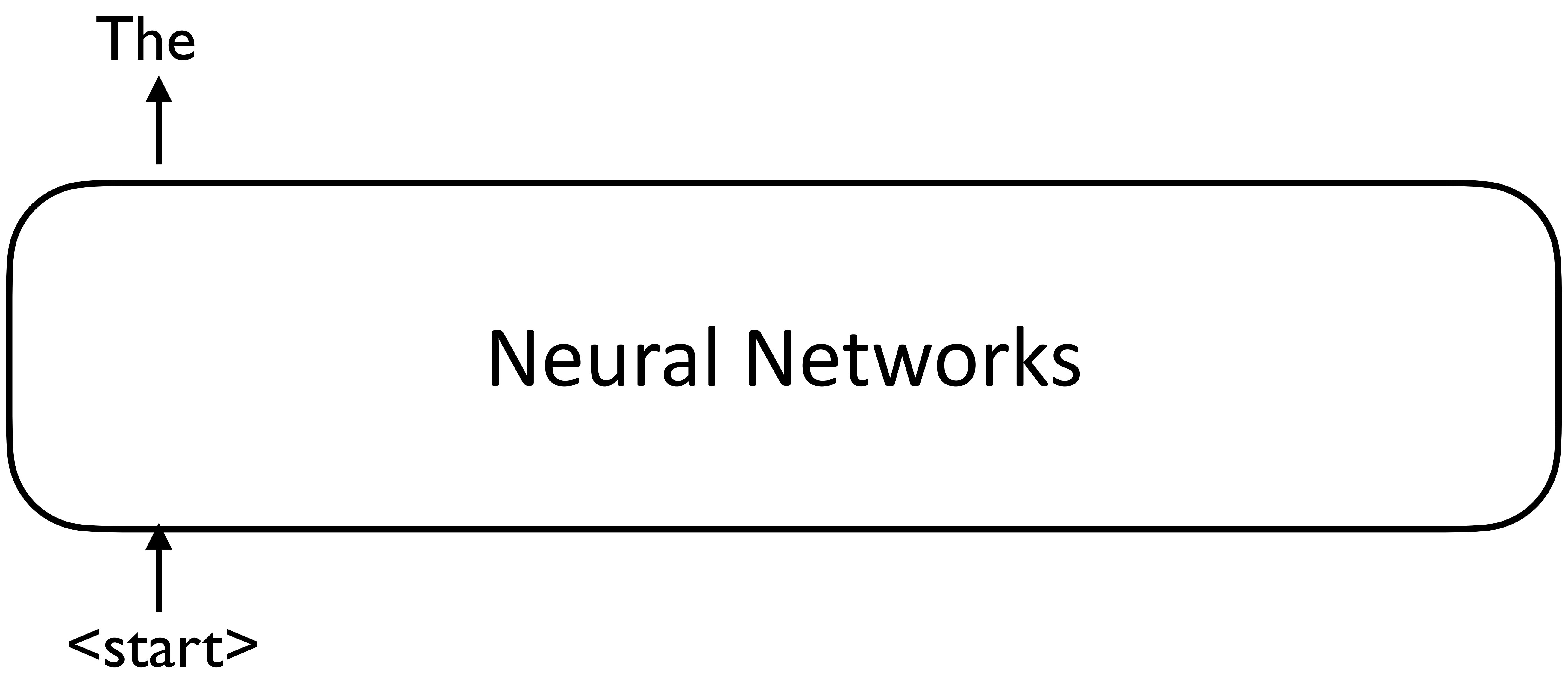
Neural Language Models

Is language modeling MLE? ✅

Are language models generative models? ✅

Can we compute p(x) given x? Can we sample new x? ✅

At inference time, to generate:



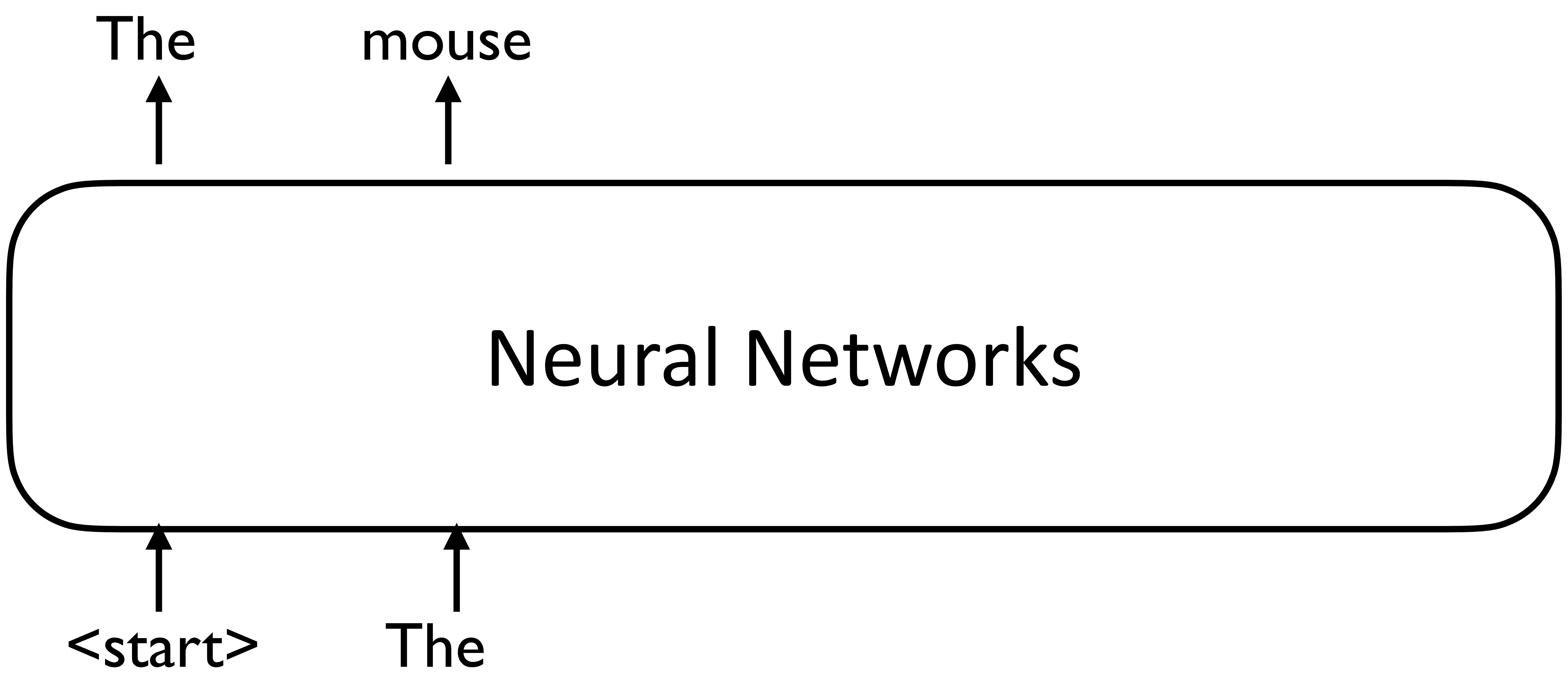
Neural Language Models

Is language modeling MLE? ✅

Are language models generative models? ✅

Can we compute p(x) given x? Can we sample new x? ✅

At inference time, to generate:



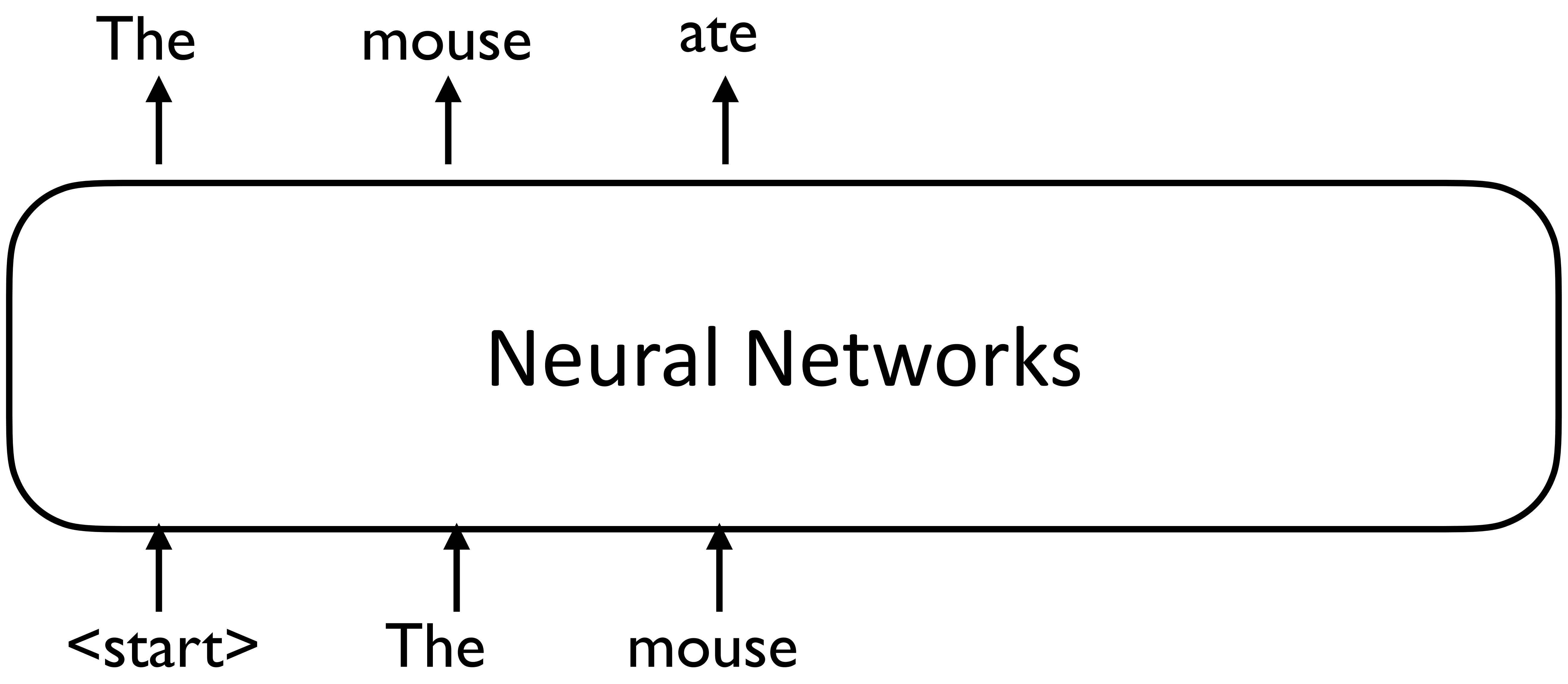
Neural Language Models

Is language modeling MLE? ✅

Are language models generative models? ✅

Can we compute p(x) given x? Can we sample new x? ✅

At inference time, to generate:



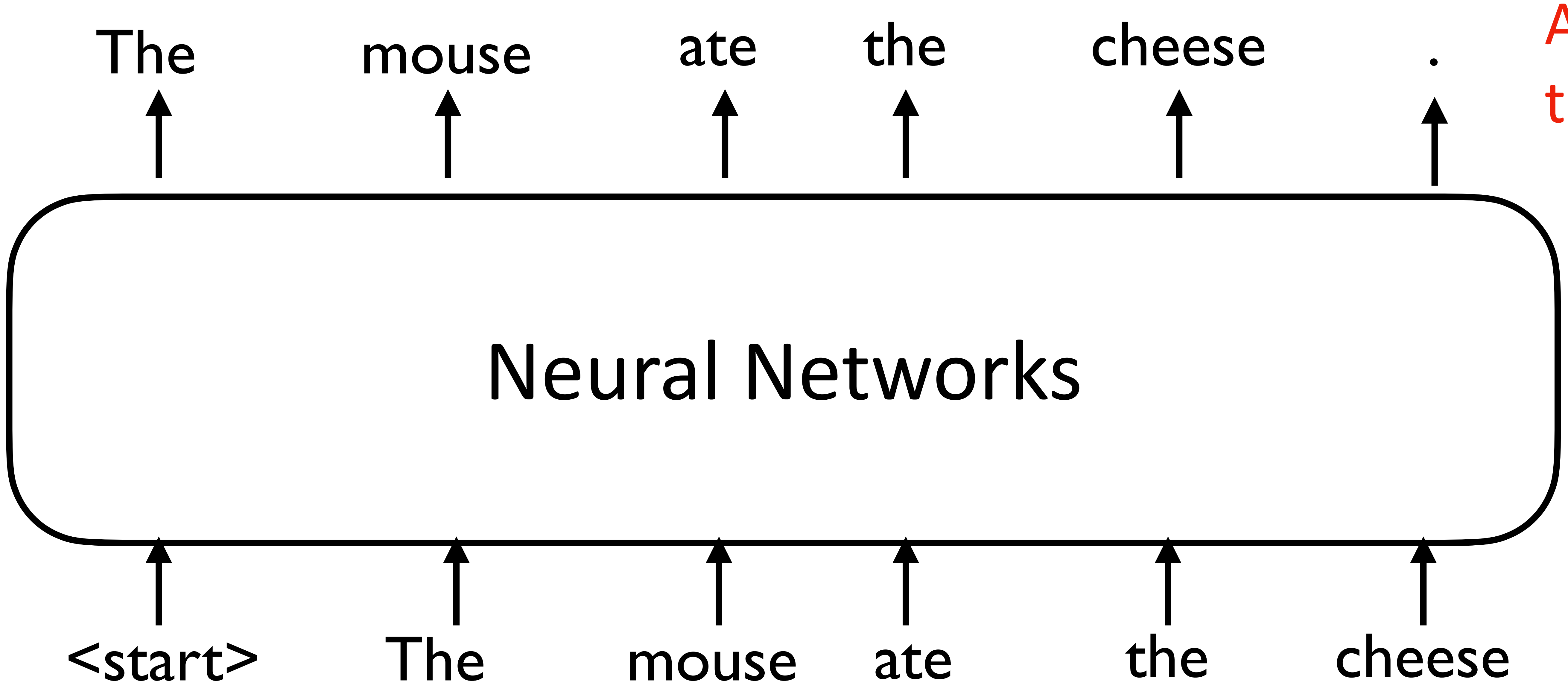
Neural Language Models

Is language modeling MLE? ✅

Are language models generative models? ✅

Can we compute p(x) given x? Can we sample new x? ✅

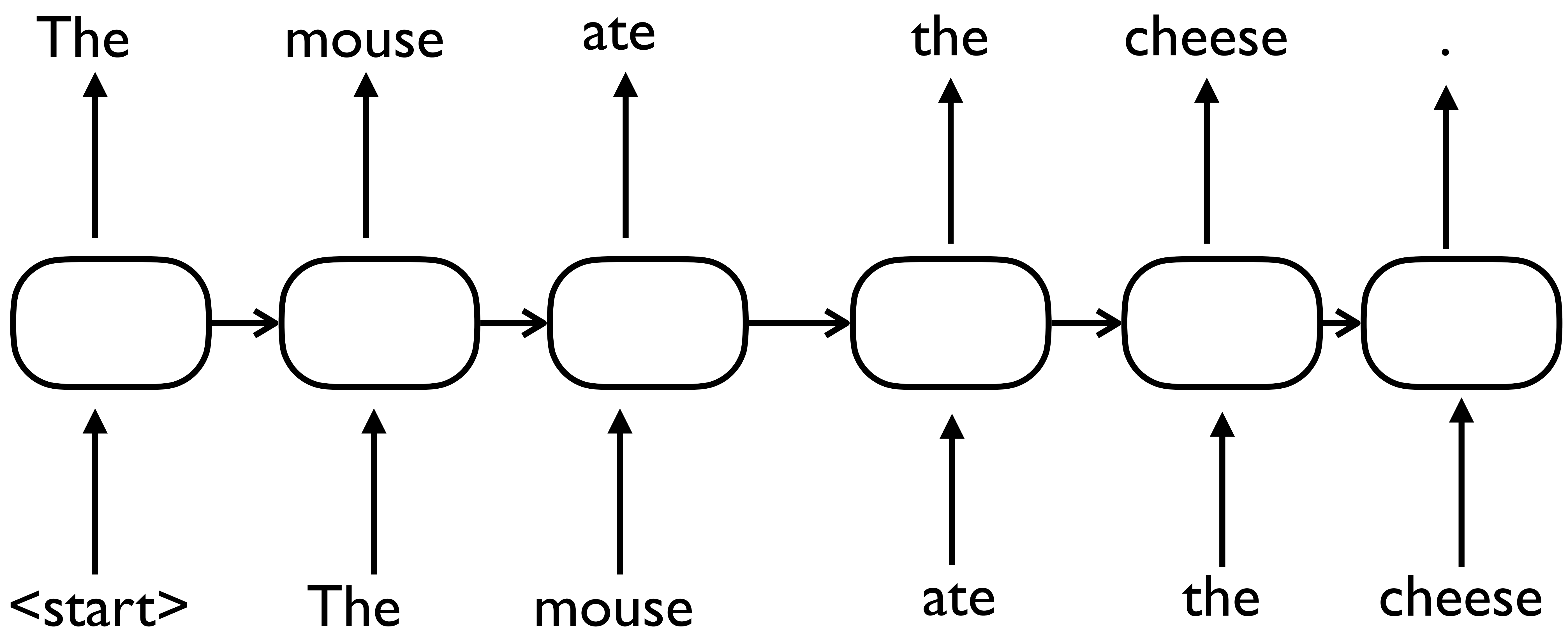
At inference time, to generate:



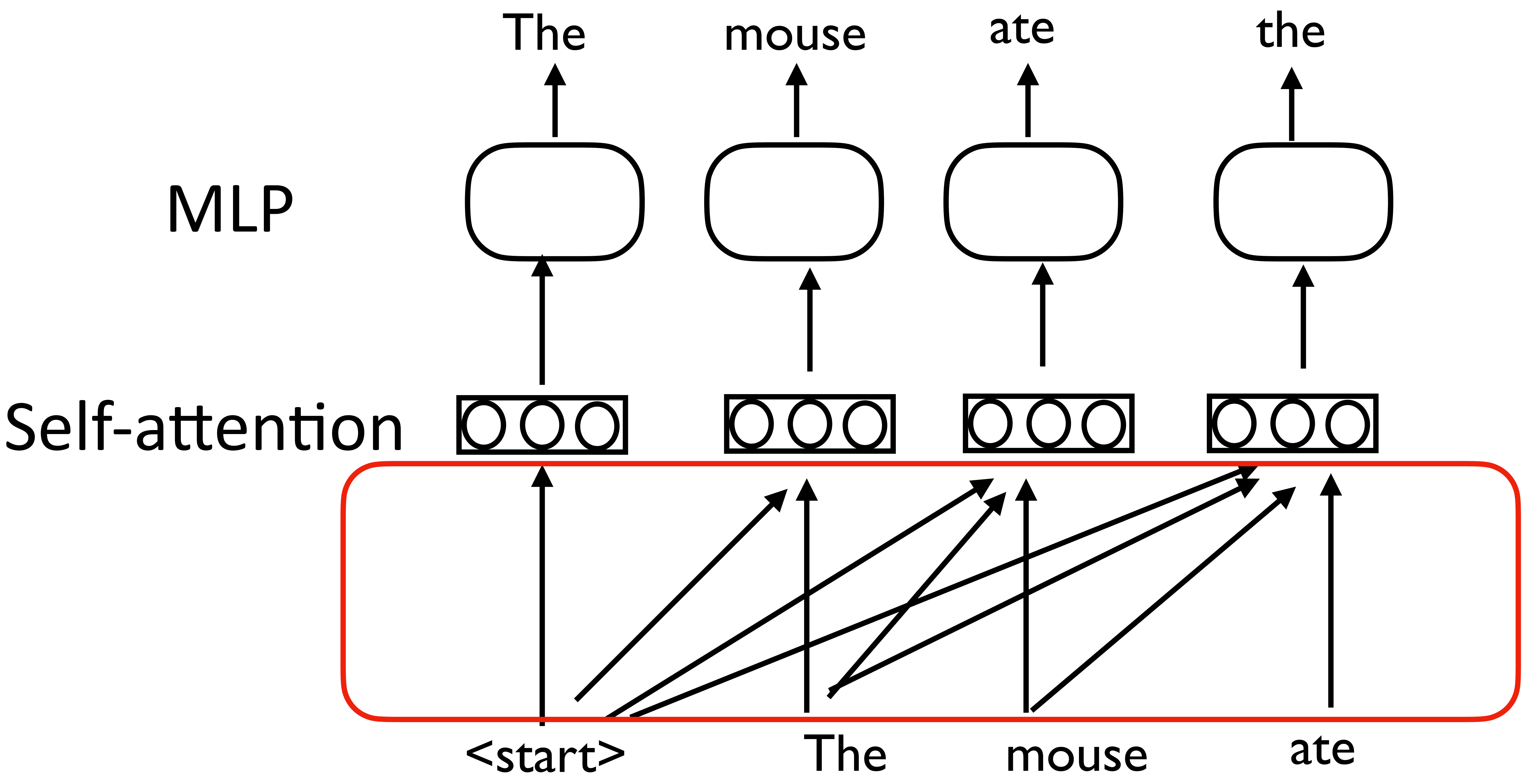
Autoregressive generation has to generate token by token

Cann't parallelize, efficiency of autoregressive decoding is still an important research topic

RNN Language Models



Transformer Language Models



Self-attention only attends to the tokens on the left (masked attention)

Neural Language Models

Language model is the fundamental block to model language distribution p(x)

For a long time, to solve specific tasks:

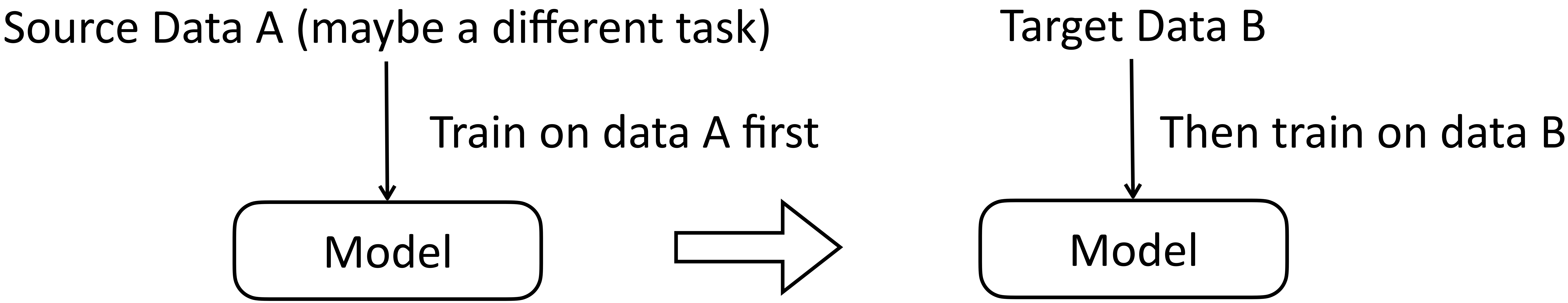
Image/text/audio → Encoder → Decoder → Text

When we have a better arch/training for LM, we can have a better decoder

Not long ago, some people think purely language models is useless because it does not directly address tasks, and LM performance may not transfer to downstream tasks

Some impactful papers are rejected by such reviewers (e.g. transformer-XL)

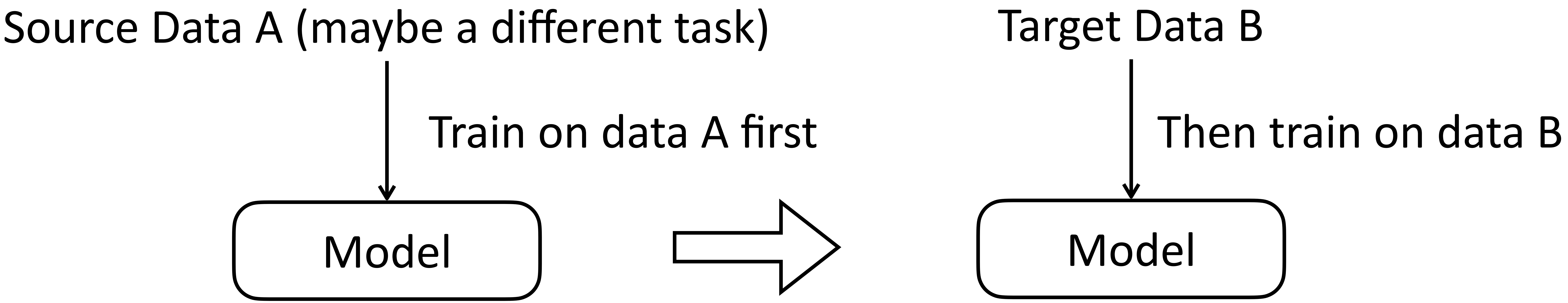
Pretraining



Classically, this is transfer Learning

It is now called pretraining because of the scale of A

Pretraining



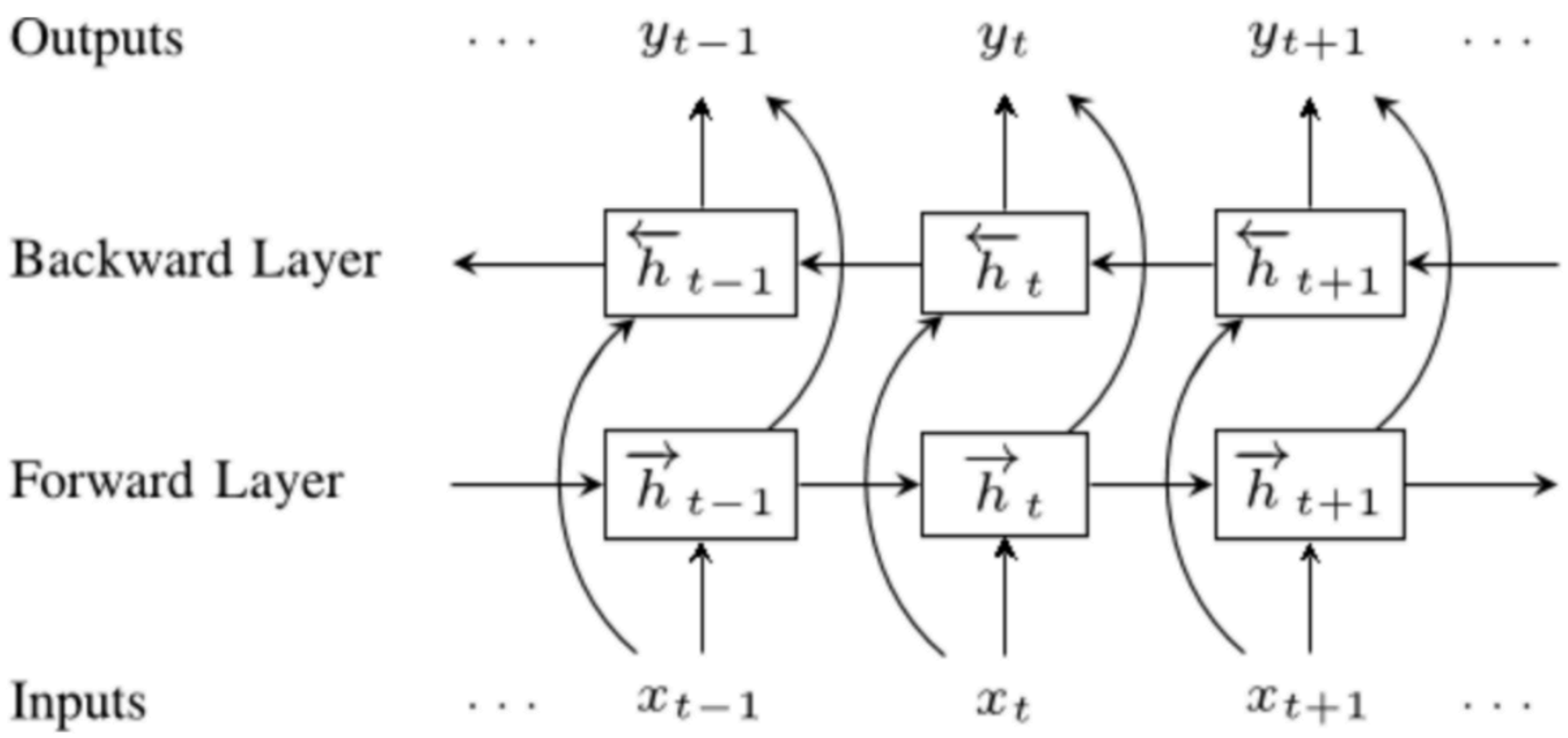
For supervised training, data A is often limited

How can we find large-scale data A to train?

ELMO

Self-supervised Pretraining

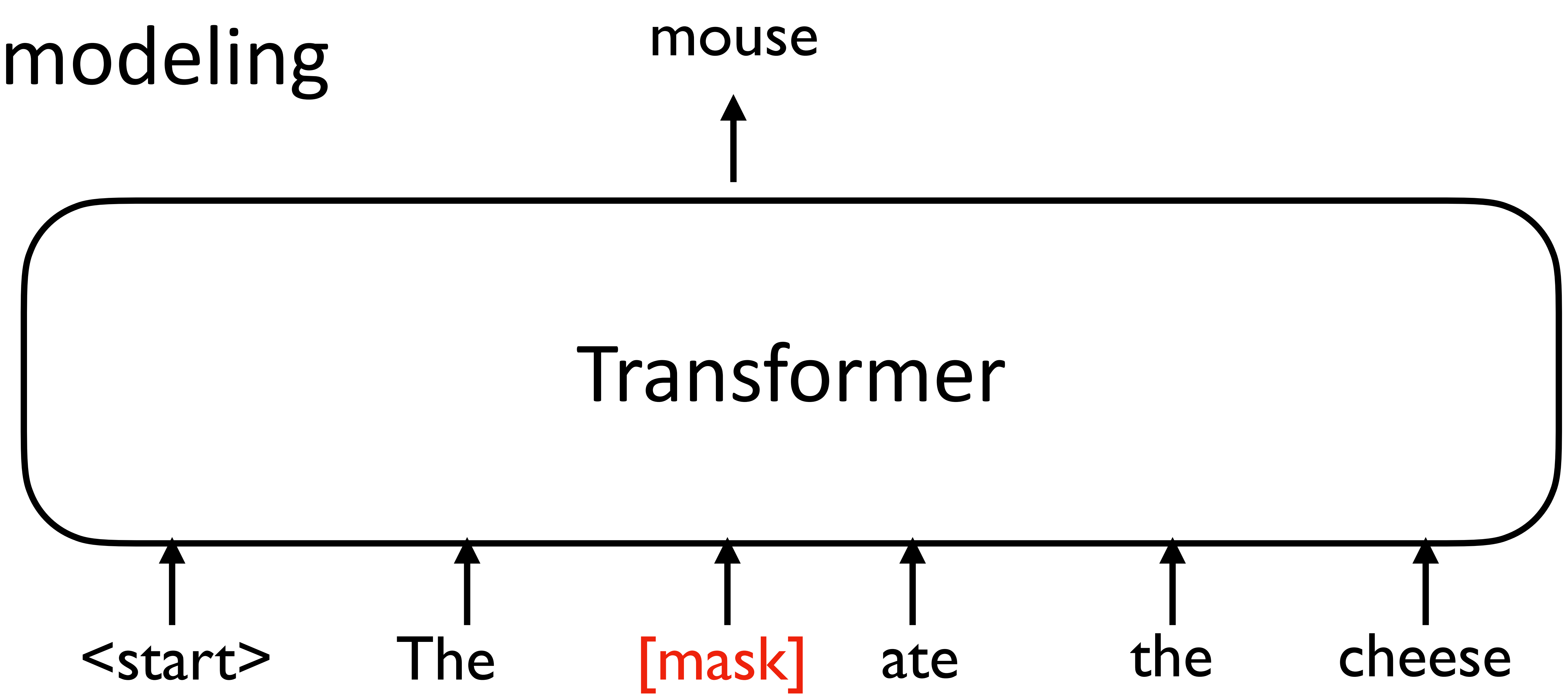
Construct supervision from unannotated data



Peters et al. Deep contextualized word representations. NAACL 2018

BERT

Mask language modeling



Construct a synthetic task from raw text only

Can be made very large-scale

Is Bert a language model? Is it a generative model?

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019.