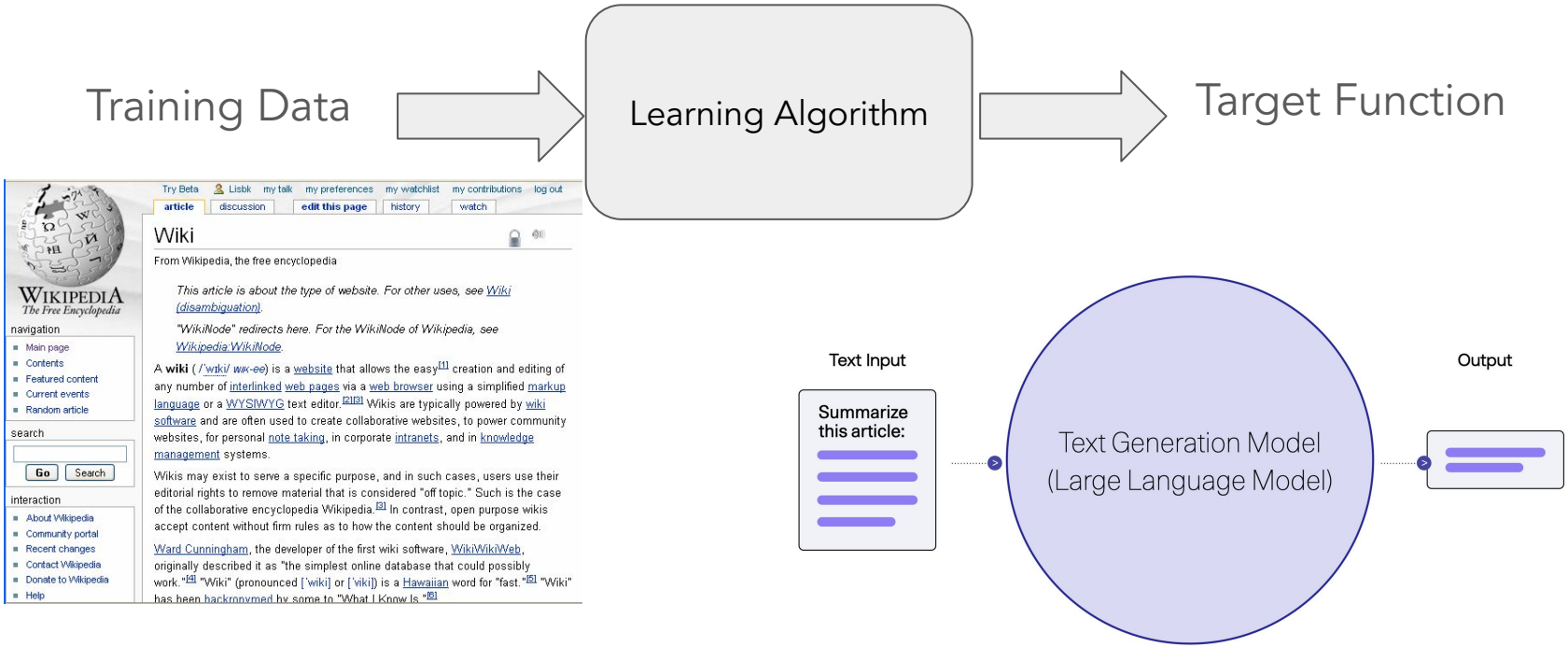
Georgia Tech
College of Computing

# CX4240 Spring 2026

# (Large) Language Model (Part II): Post-Training

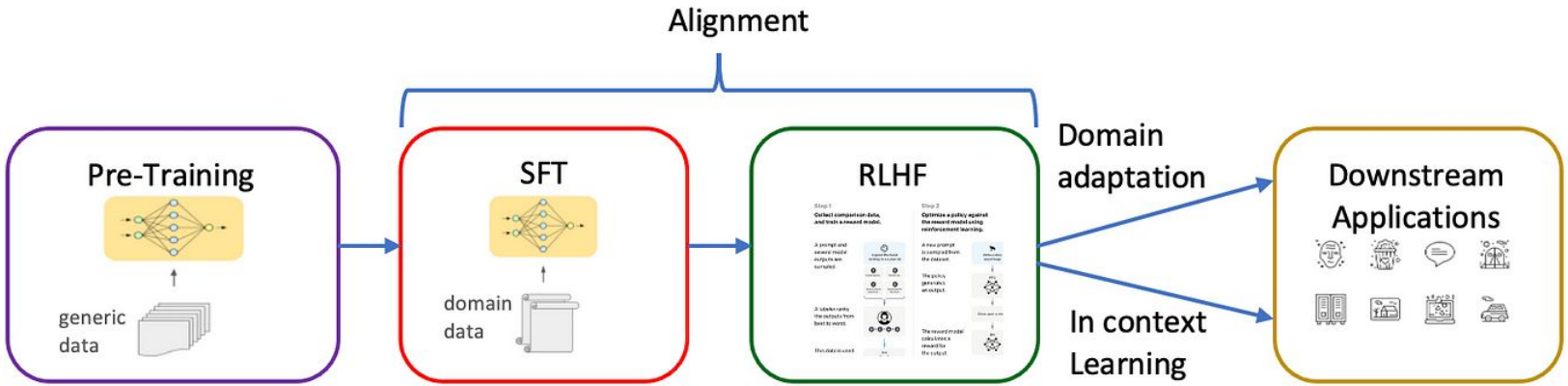
Bo Dai
School of CSE, Georgia Tech
bodai@cc.gatech.edu

# (Large) Language Models

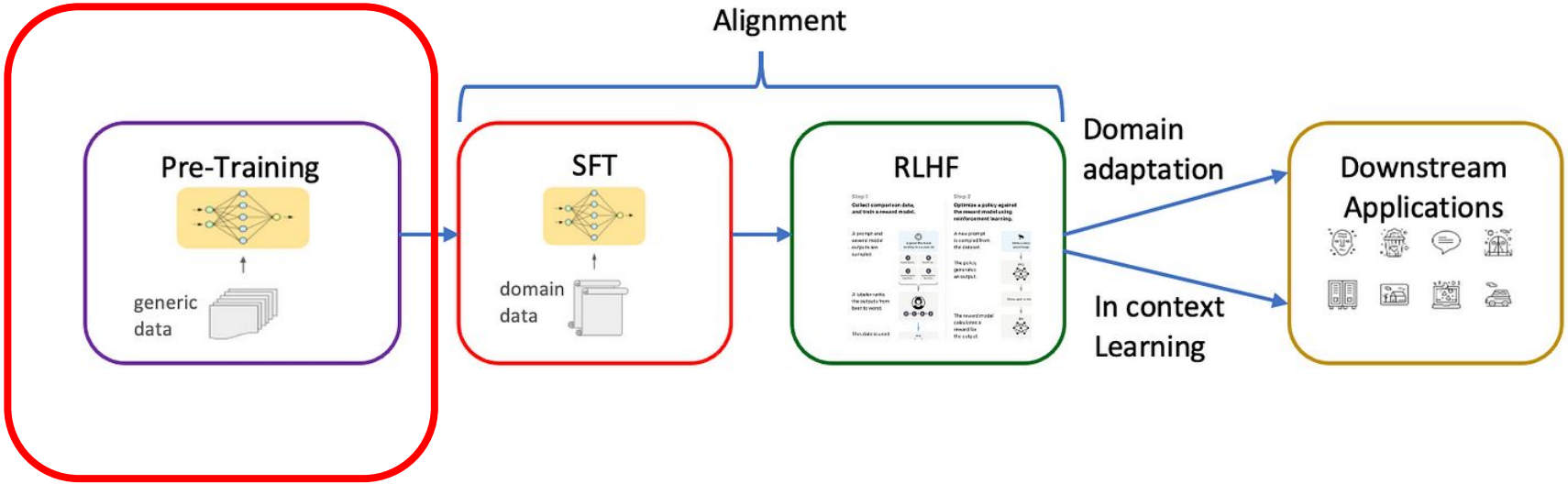



Unlabelled Data: text sequences

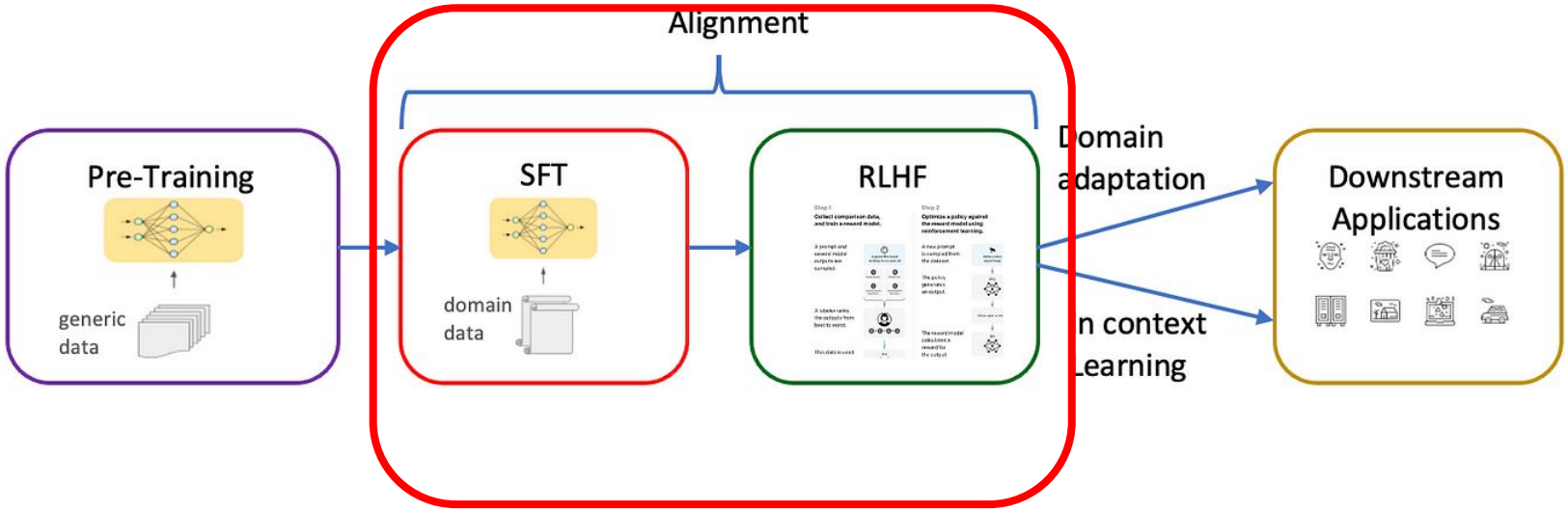
# LLM Training



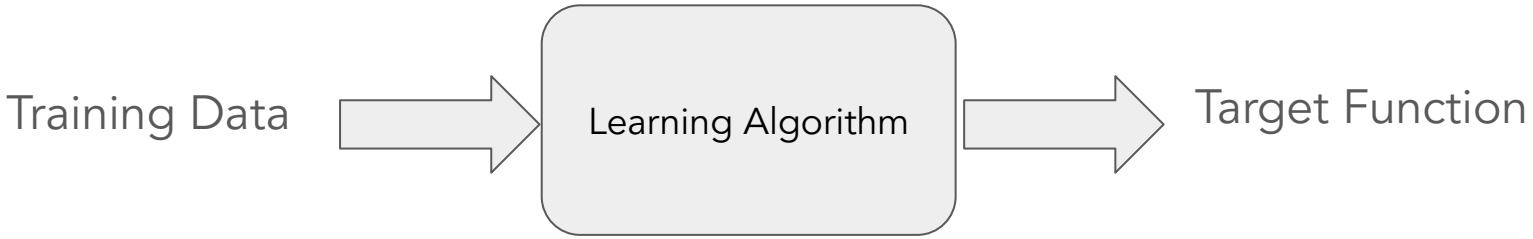
# LLM Training



# LLM Training



# Pretraining of (Large) Language Models




$$D = \{X^i\}_{i=1}^n, \quad X^i = \{x_j^i\}_{j=1}^t$$

$$p(X)$$

1. Build probabilistic models
   Categorical Distribution + Autoregressive + RNN/Transformer
2. Derive loss function (by MLE or MAP….)
   MLE
3. Select optimizer
   Stochastic Gradient Descent

# (Large) Language Models

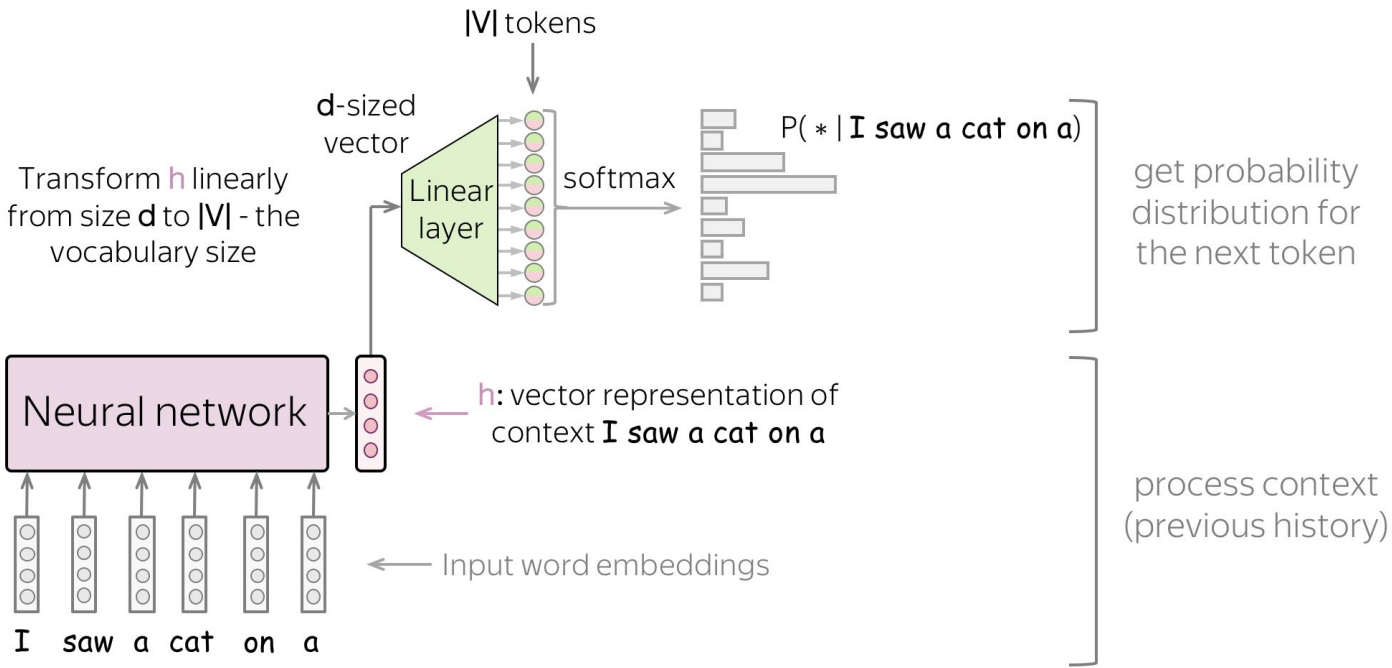
$$
\begin{aligned}
p(X) &= p\left(\{x_j\}_{j=1}^{t}\right) \\
&= \prod_{j=1}^{t} p(x_j | x_{<j}) \\
&= \prod_{j=1}^{t} \frac{\exp(W_{x_j} \phi(x_{<j}))}{\sum_{l=1}^{V} \exp(W_l \phi(x_{<j}))}
\end{aligned}
$$

# Recursive Neural Network in (Large) Language Models

$$p(X) = p\left(\{x_j\}_{j=1}^{t}\right) \qquad O(|V|^T)$$

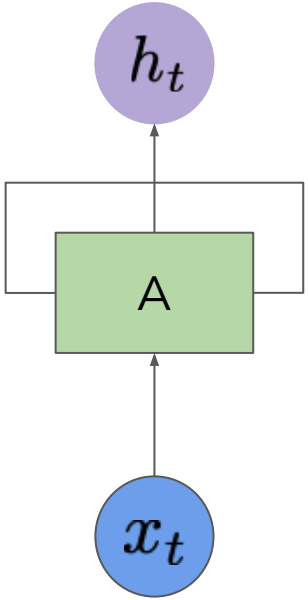
$$= \prod_{j=1}^{t} p(x_j | x_{<j})$$

$$= \prod_{j=1}^{t} \frac{\exp(W_{x_j} \phi(x_{<j}))}{\sum_{l=1}^{V} \exp(W_l \phi(x_{<j}))} \qquad \text{RNN}$$

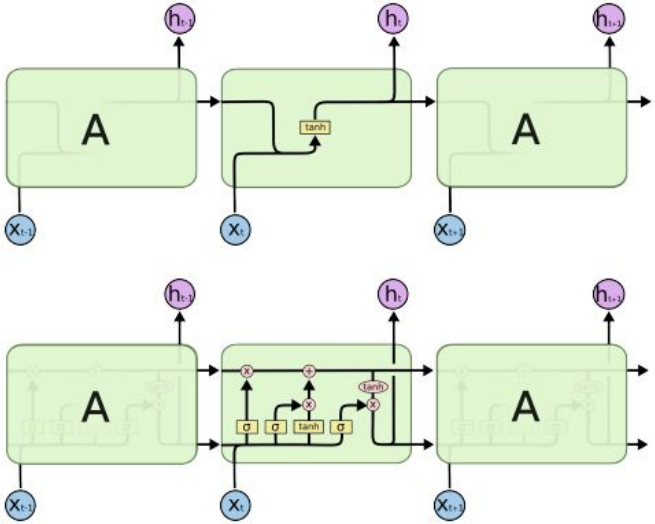



https://lena-voita.github.io/nlp_course/language_modeling.html

# RNN Cell



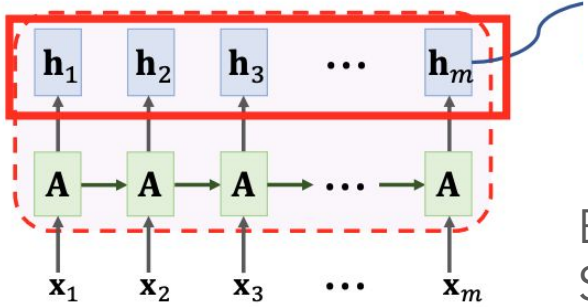

$$\frac{\exp(W_{x_j}\phi(x_{<j}))}{\sum_{l=1}^{V}\exp(W_l\phi(x_{<j}))}$$




Simple RNN

LSTM

# Bottleneck in RNN



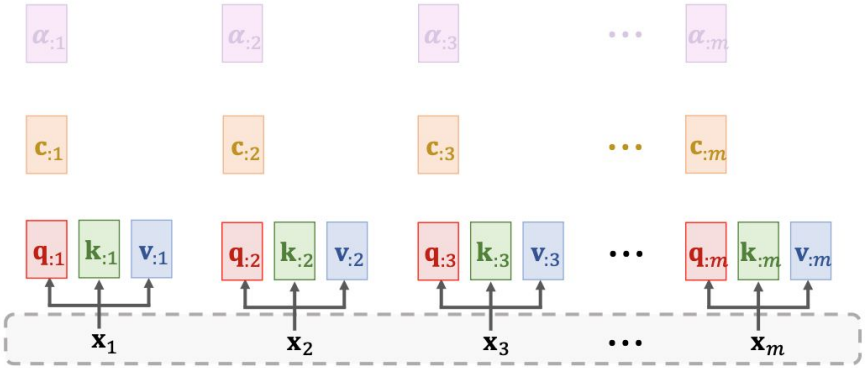

$$h_j = \phi(x_{<j})$$

$$\frac{\exp(W_{x_j}\phi(x_{<j}))}{\sum_{l=1}^{V}\exp(W_l\phi(x_{<j}))}$$

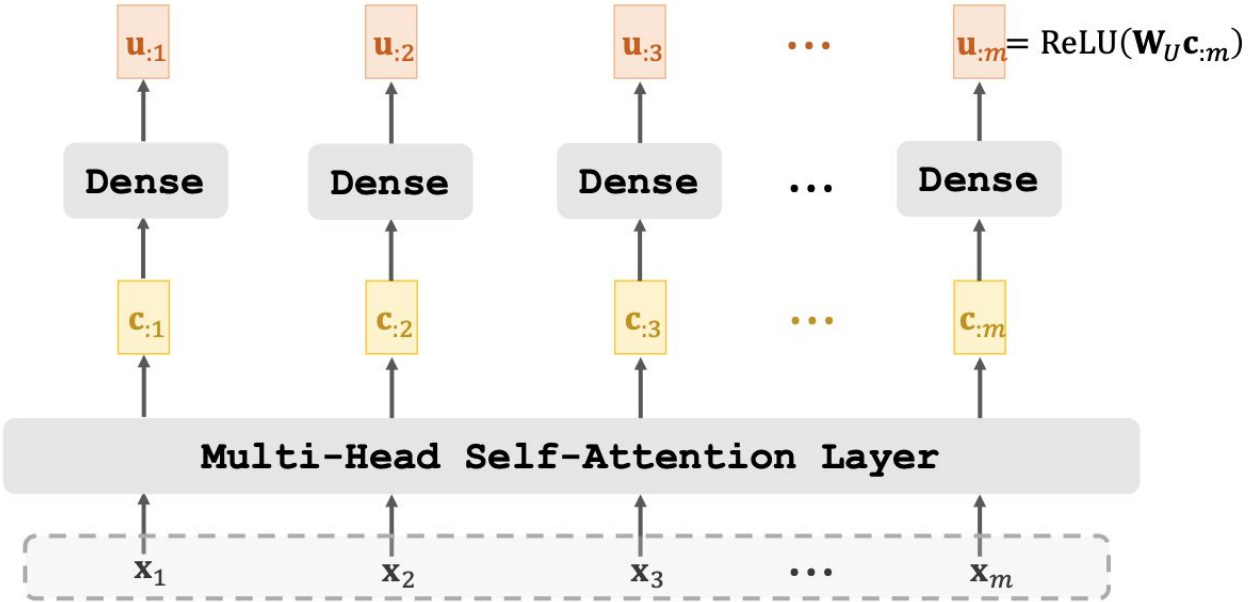
Bottleneck:
Sequences bottlenecked through a fixed-sized vector.

# Self-Attention

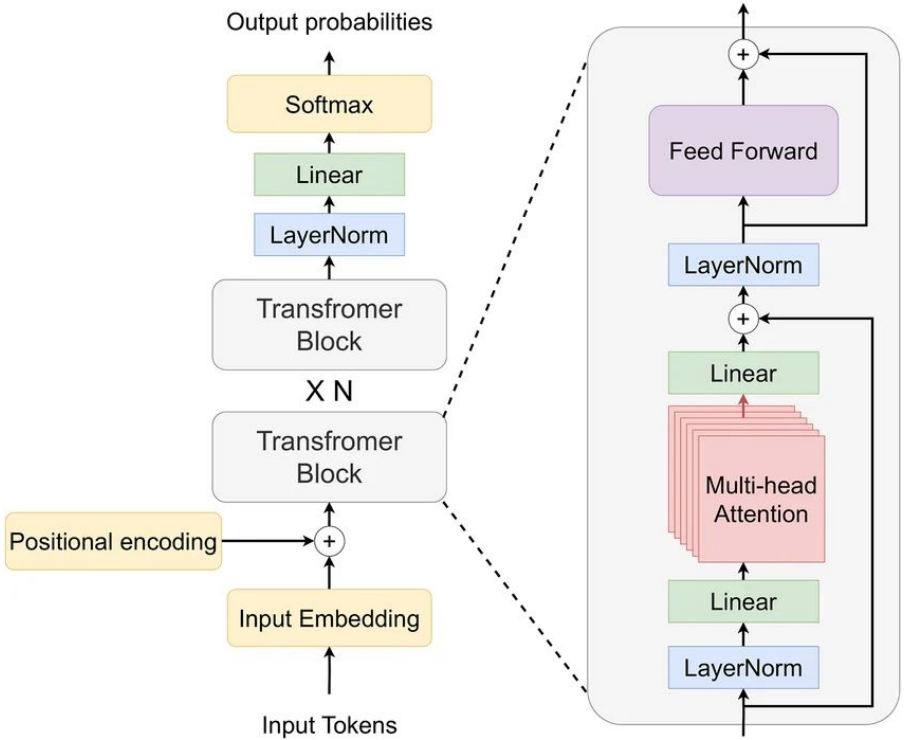
**Context vector:** $\mathbf{c}_{:1} = \alpha_{11}\mathbf{v}_{:1} + \cdots + \alpha_{m1}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:1}.$



# Multi-Headed Self-Attention



# Transformer: Multi-Layer Multi-Headed Self-Attention!



# Is Pretraining Enough for (L)LMs?

What we have:

$$p(X) = p\left(\{x_j\}_{j=1}^{t}\right) = \prod_{j=1}^{t} p(x_j | x_{<j})$$

Generating texts unconditionally

# Is Pretraining Enough for (L)LMs?

What we have:

$$p(X) = p\left(\{x_j\}_{j=1}^{t}\right) = \prod_{j=1}^{t} p(x_j | x_{<j})$$

Generating texts unconditionally

Answer Question? Generating texts condition on answer

# Is Pretraining Enough for (L)LMs?

What we have:

$$p(X) = p\left(\{x_j\}_{j=1}^{t}\right) = \prod_{j=1}^{t} p(x_j | x_{<j})$$

Generating texts unconditionally

Answer Question? Generating texts condition on answer

$$p(Y|X) = \prod_{l=1}^{L} p(y_l \mid y_{<l}, X)$$

# Is Pretraining Enough for (L)LMs?

What we have:

$$p(X) = p\left(\{x_j\}_{j=1}^{t}\right) = \prod_{j=1}^{t} p(x_j | x_{<j})$$

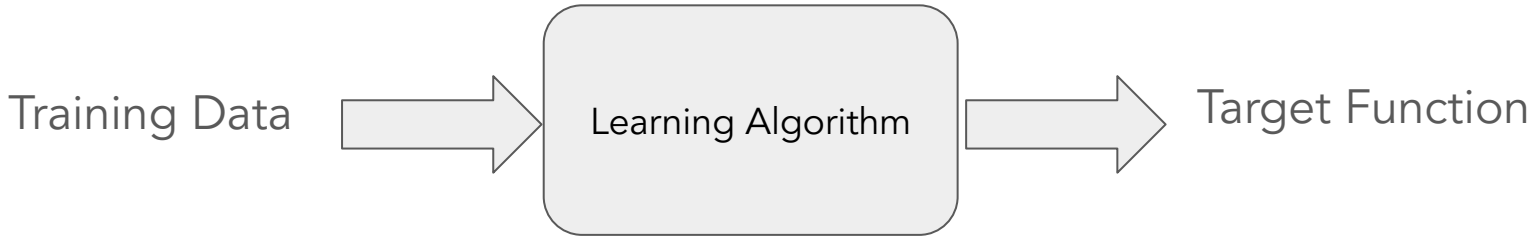
Generating texts unconditionally

Answer Question? Generating texts condition on answer

$$p(Y|X) = \prod_{l=1}^{L} p(y_l \mid y_{<l}, X)$$

Never be trained

# Supervised Fine-tuning (Instruction Fine-tuning)

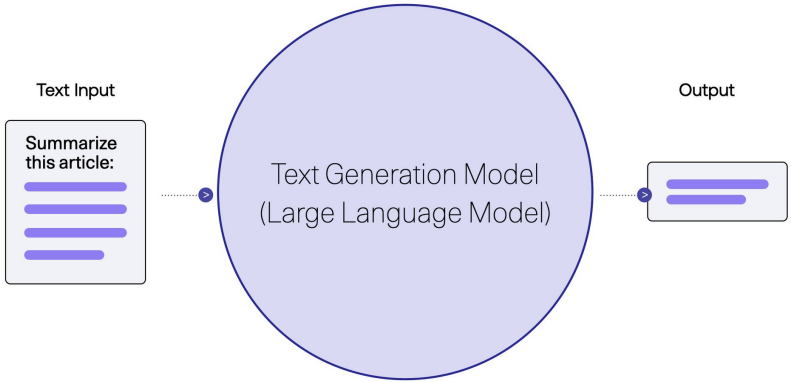



$$D = \{X^i, Y^i\}_{i=1}^{n}, \quad X^i = \{x_j^i\}_{j=1}^{t}, \; Y^i = \{y_l^i\}_{l=1}^{L}$$

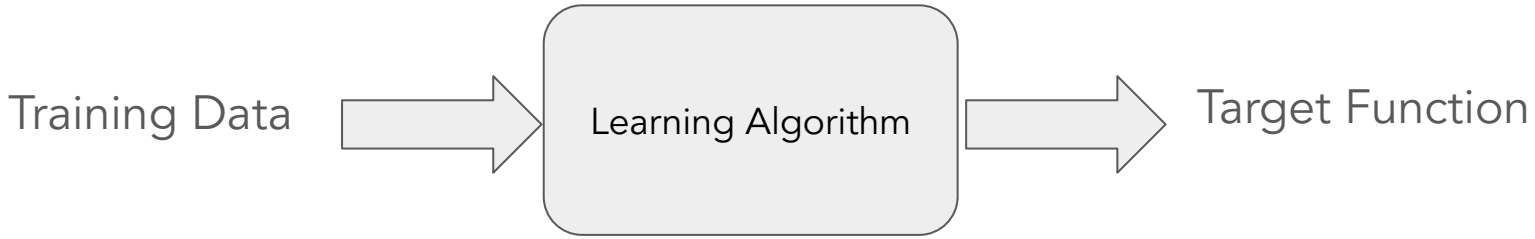
$$p(Y|X)$$

**MATH Dataset (Ours)**

**Problem:** Tom has a red marble, a green marble, a blue marble, and three identical yellow marbles. How many different groups of two marbles can Tom choose?
**Solution:** There are two cases here: either Tom chooses two yellow marbles (1 result), or he chooses two marbles of different colors ($\binom{4}{2} = 6$ results). The total number of distinct pairs of marbles Tom can choose is $1 + 6 = \boxed{7}$.
**Problem:** If $\sum_{n=0}^{\infty} \cos^{2n} \theta = 5$, what is $\cos 2\theta$?
**Solution:** This geometric series is $1 + \cos^2 \theta + \cos^4 \theta + \cdots = \frac{1}{1-\cos^2 \theta} = 5$. Hence, $\cos^2 \theta = \frac{4}{5}$. Then $\cos 2\theta = 2\cos^2 \theta - 1 = \boxed{\frac{3}{5}}$.
**Problem:** The equation $x^2 + 2x = i$ has two complex solutions. Determine the product of their real parts.
**Solution:** Complete the square by adding 1 to each side. Then $(x+1)^2 = 1 + i = e^{\frac{i\pi}{4}}\sqrt{2}$, so $x + 1 = \pm e^{\frac{i\pi}{8}}\sqrt[4]{2}$. The desired product is then $\left(-1 + \cos\left(\frac{\pi}{8}\right)\sqrt[4]{2}\right)\left(-1 - \cos\left(\frac{\pi}{8}\right)\sqrt[4]{2}\right) =$ $1 - \cos^2\left(\frac{\pi}{8}\right)\sqrt{2} = 1 - \frac{\left(1+\cos\left(\frac{\pi}{4}\right)\right)}{2}\sqrt{2} = \boxed{\frac{1-\sqrt{2}}{2}}$.



# Supervised Fine-tuning (Instruction Fine-tuning)




$$D = \{X^i, Y^i\}_{i=1}^{n}, \quad X^i = \{x_j^i\}_{j=1}^{t}, \; Y^i = \{y_l^i\}_{l=1}^{L}$$

$$p(Y|X)$$

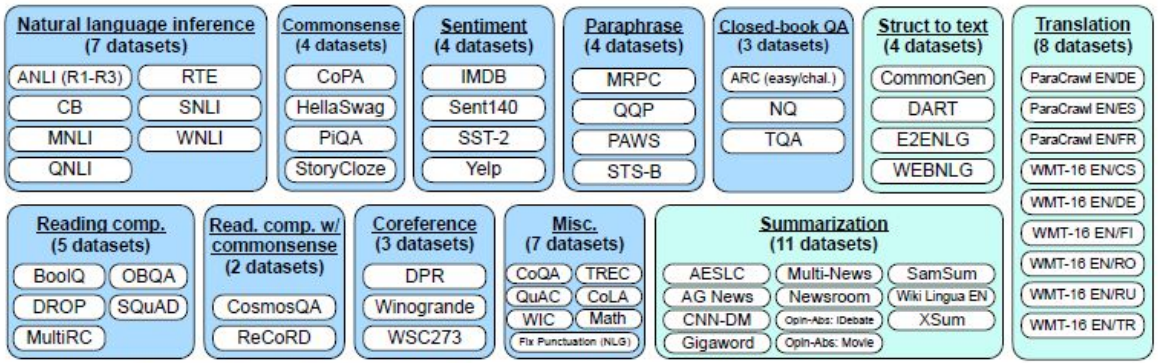
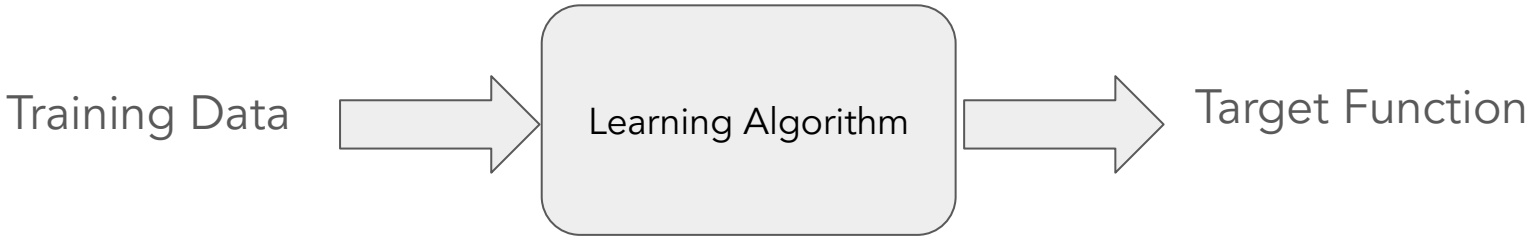



Figure 3: Datasets and task clusters used in this paper (NLU tasks in blue; NLG tasks in teal).

# Supervised Fine-tuning (Instruction Fine-tuning)

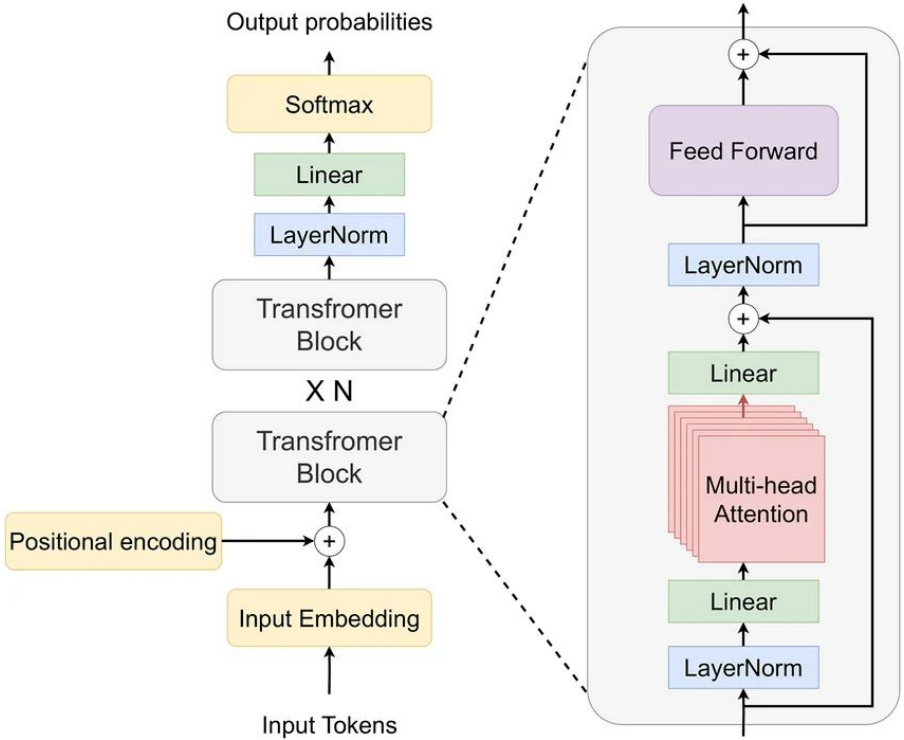



$$D = \{X^i, Y^i\}_{i=1}^n, \quad X^i = \{x_j^i\}_{j=1}^t, \; Y^i = \{y_l^i\}_{l=1}^L$$

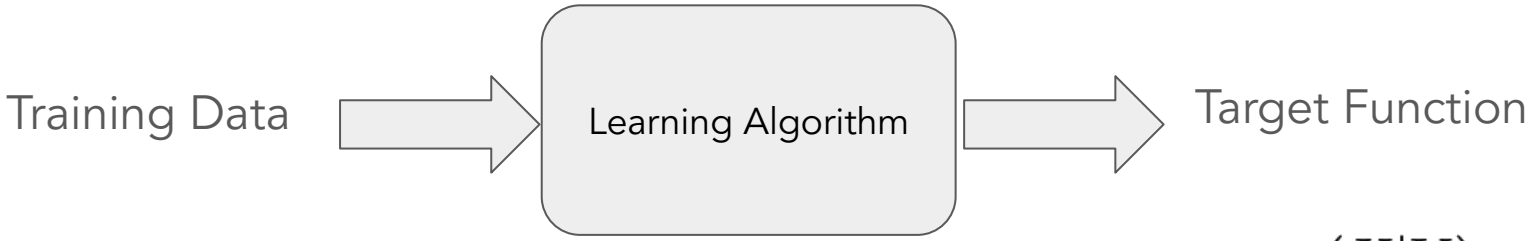
$$p(Y|X)$$

1. Build probabilistic models
   Fixed the same as in pretraining:
   Categorical Distribution + Autoregressive + RNN/Transformer
2. Derive loss function (by MLE or MAP….)
3. Select optimizer

# Transformer: Multi-Layer Multi-Headed Self-Attention!



# Supervised Fine-tuning (Instruction Fine-tuning)




$D = \{X^i, Y^i\}_{i=1}^n, \quad X^i = \{x_j^i\}_{j=1}^t, \; Y^i = \{y_l^i\}_{l=1}^L$

$p(X|Y)$

1. Build probabilistic models
2. Derive loss function (by MLE or MAP….)
   MLE
3. Select optimizer

# MLE for Supervised Fine-tuning (SFT)

- Given all input data $D = \{X^i, Y^i\}_{i=1}^n, \quad X^i = \{x_j^i\}_{j=1}^t, \; Y^i = \{y_l^i\}_{l=1}^L$

$$p(Y \mid X) = \prod_{l=1}^{L} p(y_l \mid y_{<l}, X)$$

- Log-likelihood

$$\begin{aligned}
\ell(\phi) &= \sum_{i=1}^{n} p(Y^i \mid X^i; \phi) = \sum_{i=1}^{n} \sum_{l=1}^{L} \log \left(y_l^i \mid y_{<l}^i, X^i; \phi\right) \\
&= \sum_{i=1}^{n} \sum_{l=1}^{L} \log \frac{\exp\left(W_{y_l^i} \phi\left(y_{<l}^i \mid X^i\right)\right)}{\sum_{v=1}^{V} \exp\left(W_v \phi\left(y_{<l}^i \mid X^i\right)\right)} \\
&= \sum_{i=1}^{n} \sum_{l=1}^{L} W_{y_l^i} \phi\left(y_{<l}^i \mid X^i\right) - \sum_{i=1}^{n} \sum_{j=1}^{t} \log \sum_{v=1}^{V} \exp\left(W_v \phi\left(y_{<l}^i \mid X^i\right)\right)
\end{aligned}$$

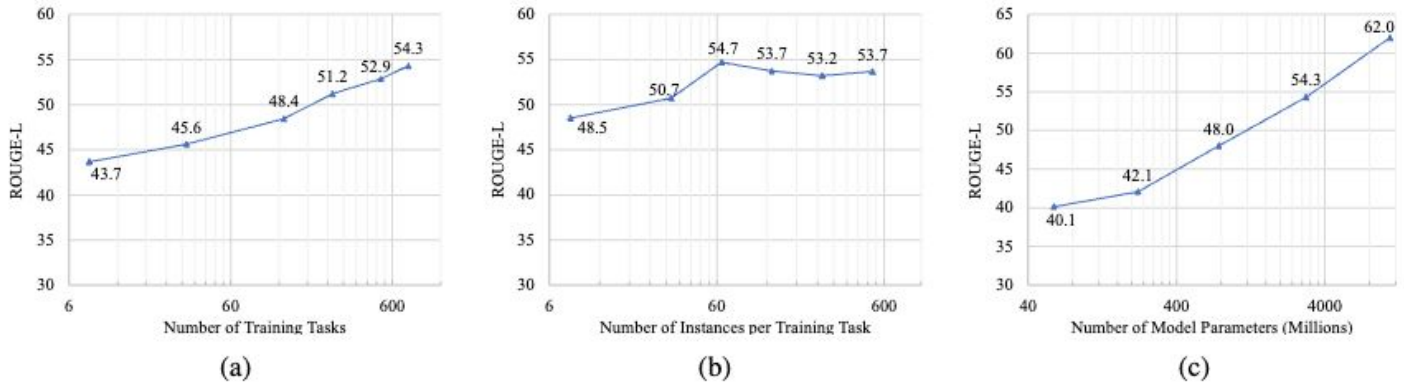
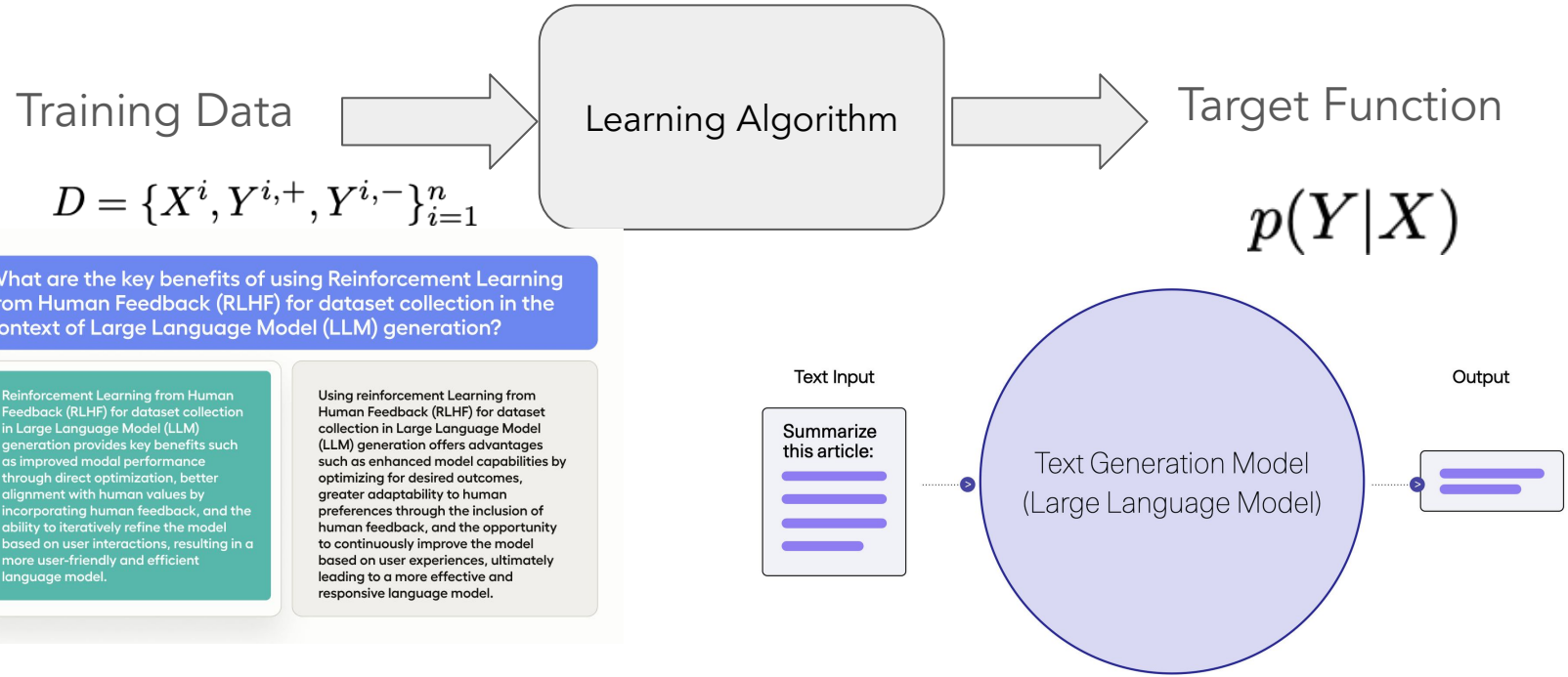



Figure 5: Scaling trends of models performance (§7.1) as a function of (a) the number of training tasks; (b) the number of instances per training task; (c) model sizes. $x$-axes are in log scale. The **linear growth of model performance with exponential increase in observed tasks and model size** is a promising trend. Evidently, the performance gain from more instances is limited.

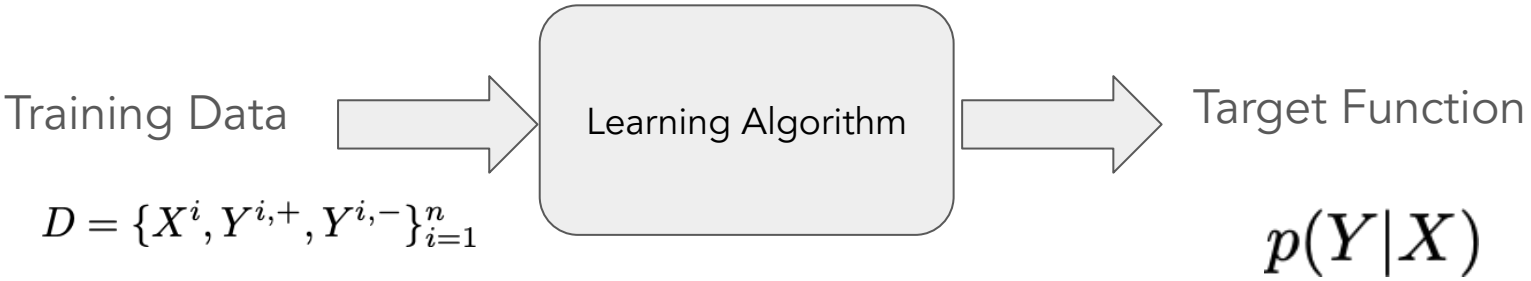
Wang, Yizhong, et al. "Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks." *arXiv preprint arXiv:2204.07705* (2022).

# RLHF for (Large) Language Models



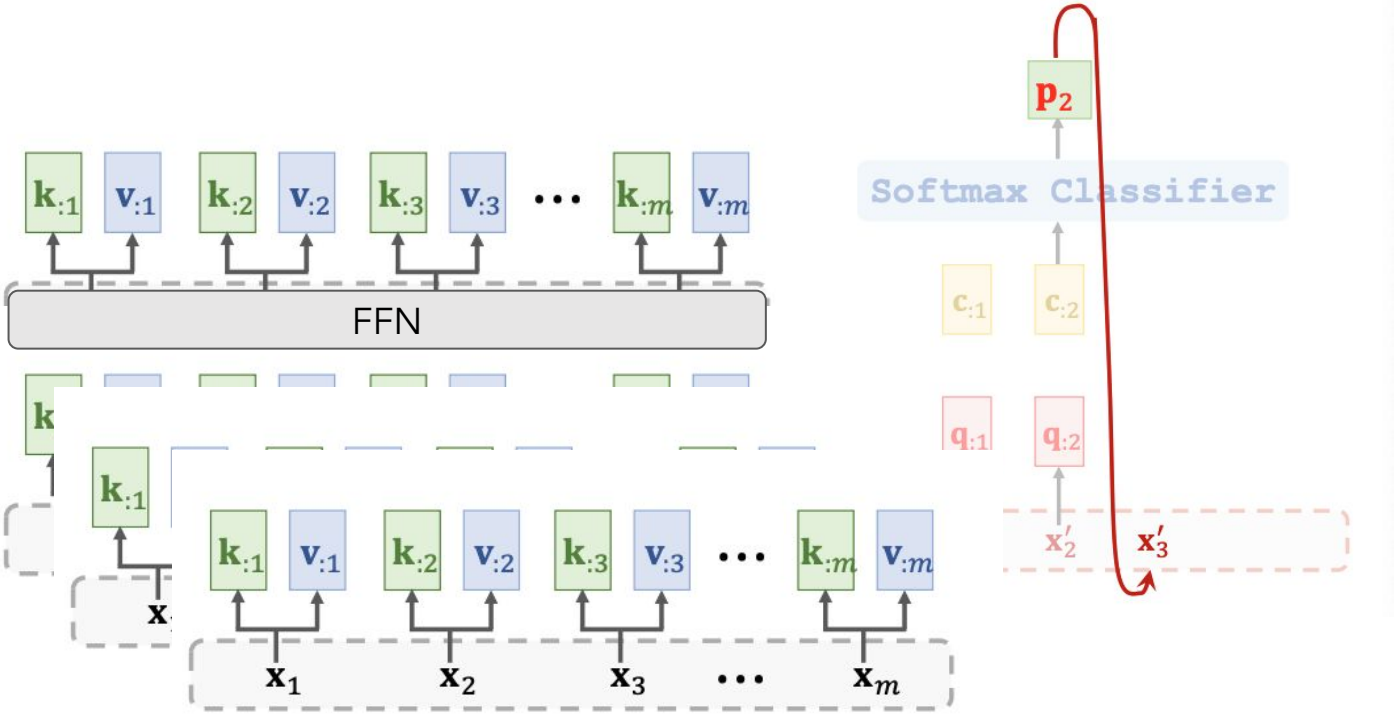

Preference Data: Prompt & Positive/ Negative Answers

# RLHF for (Large) Language Models

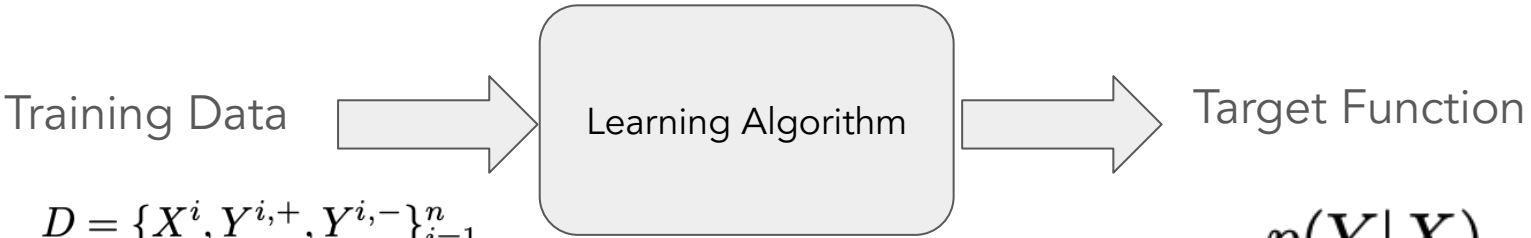



1. Build probabilistic models
   Fixed the same as in pretraining:
   Categorical Distribution + Autoregressive + RNN/Transformer
   + classification head (only for learning)
2. Derive loss function (by MLE or MAP….)
3. Select optimizer

# Transformer: Multi-Headed Multi-Layer Parallel Attention!



# RLHF for (Large) Language Models



# RLHF for (Large) Language Models

Training Data → Learning Algorithm → Target Function

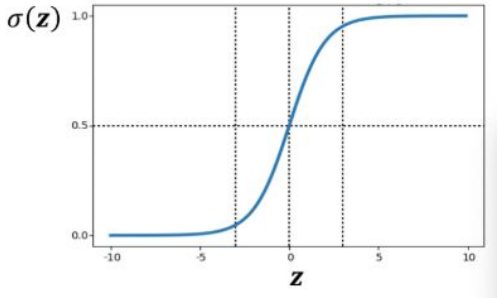
$$D = \{X^i, Y^{i,+}, Y^{i,-}\}_{i=1}^n$$

$$\{X^i, Z^{i,1}, Z^{i,2}, y^i\}_{i=1}^n$$

$$y^i = \mathbf{1}(Z^{i,1} \succ Z^{i,2})$$

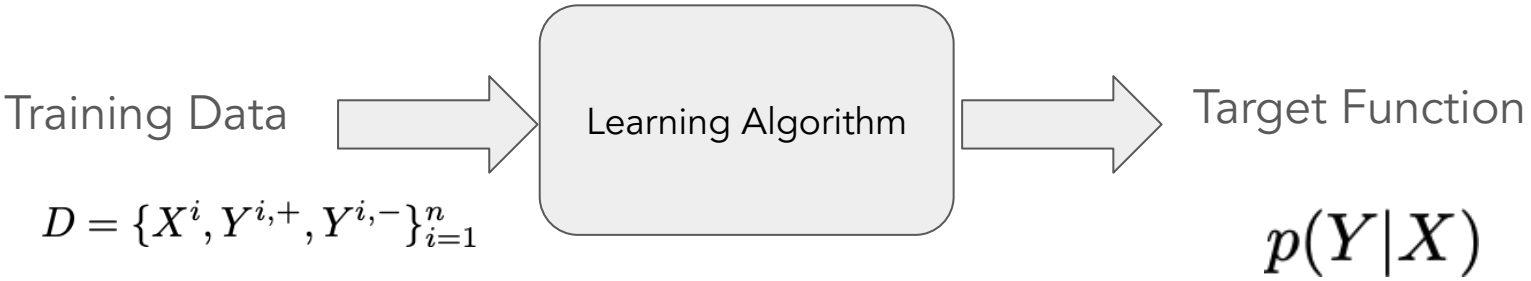
$$p(Y|X)$$

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$$




$$p(Z^1 \succ Z^2|X) = \sigma\left(\log \frac{p(Z^1|x)}{p_{\text{ref}}(Z^1|x)} - \log \frac{p(Z^2|x)}{p_{\text{ref}}(Z^2|x)}\right)$$

# RLHF for (Large) Language Models




1. Build probabilistic models
2. Derive loss function (by MLE or MAP….)
   MLE
3. Select optimizer

# MLE for Preference Learning - Direct Preference Optimization (DPO)

- Given all input data $D = \{X^i, Y^{i,+}, Y^{i,-}\}_{i=1}^n$

  $\{X^i, Z^{i,1}, Z^{i,2}, y^i\}_{i=1}^n$ ⇨ $p(Z^1 \succ Z^2|X) = \sigma\left(\log \frac{p(Z^1|x)}{p_{\text{ref}}(Z^1|x)} - \log \frac{p(Z^2|x)}{p_{\text{ref}}(Z^2|x)}\right)$

  $y^i = \mathbf{1}(Z^{i,1} \succ Z^{i,2})$
- Log-likelihood

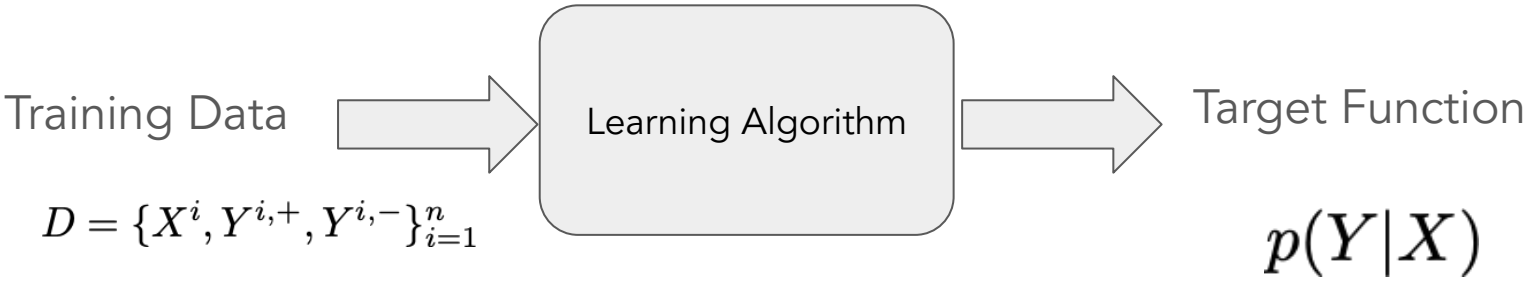
$$\begin{aligned}\ell(\phi) &= \sum_{i=1}^{n} y^i \log p(Z^{i,1} \succ Z^{i,2} \mid X^i; \phi) \\ &= \sum_{i=1}^{n} \log \sigma \left(\log \frac{p_\phi\left(Y^{i,+} \mid X^i\right)}{p_{\text{ref}}\left(Y^{i,+} \mid X^i\right)} - \log \frac{p_\phi\left(Y^{i,-} \mid X^i\right)}{p_{\text{ref}}\left(Y^{i,-} \mid X^i\right)}\right)\end{aligned}$$

# Online vs. Offline Samples

$$\ell(\phi) = \sum_{i=1}^{n} y^i \log p(Z^{i,1} \succ Z^{i,2} \mid X^i; \phi)$$
$$= \sum_{i=1}^{n} \log \sigma \left( \log \frac{p_\phi \left(Y^{i,+} \mid X^i\right)}{p_{\text{ref}} \left(Y^{i,+} \mid X^i\right)} - \log \frac{p_\phi \left(Y^{i,-} \mid X^i\right)}{p_{\text{ref}} \left(Y^{i,-} \mid X^i\right)} \right)$$

Where the samples Y comes from?

# RLHF for (Large) Language Models




1. Build probabilistic models
2. Derive loss function (by MLE or MAP….)
3. Select optimizer

   Stochastic Gradient Descent

# Proximal Policy Optimization (PPO) vs. DPO

1, Learn a reward model

$$\text{loss}\left(\theta\right) = -\frac{1}{\binom{K}{2}} E_{\left(x, y_w, y_l\right) \sim D}\left[\log\left(\sigma\left(r_\theta\left(x, y_w\right) - r_\theta\left(x, y_l\right)\right)\right)\right]$$

2, Policy Gradient

$$\max_{p(Y|X)} \sum_{X \sim D} \left( \mathbb{E}_{p(Y|X)}[r(X,Y)] - \lambda KL(p(Y|X)||p_{\text{ref}}(Y|X)) \right)$$

# Proximal Policy Optimization (PPO) vs. DPO

1, Learn a reward model

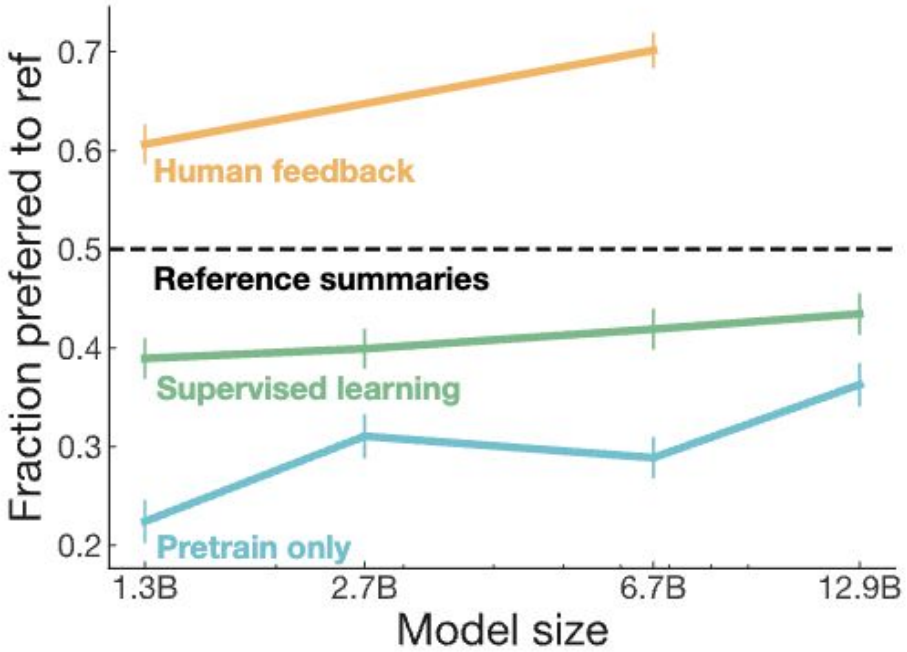
$$E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x,y_w\right)-r_\theta\left(x,y_l\right)\right)\right)\right]$$

2, Policy Gradient

$$\max_{p(Y|X)}\sum_{X\sim D}\left(\mathbb{E}_{p(Y|X)}[r(X,Y)]-\lambda KL(p(Y|X)||p_{\text{ref}}(Y|X))\right)$$

Naturally Online!

# RLHF Performance

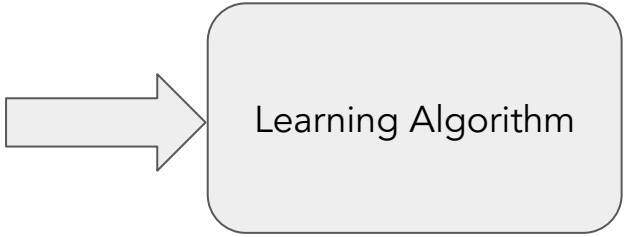



Stiennon, Nisan, et al. "Learning to summarize with human feedback." *Advances in neural information processing systems* 33 (2020): 3008-3021.

# Summary

- Pretraining of LLM is not enough
- Post-training of LLM is necessary:
  - Supervised Fine-tuning
  - Reinforcement Learning from Human Feedback
    - Online vs Offline data

# Supervised Learning vs. Unsupervised Learning




Training Data

Supervised $\{x^i, y^i\}_{i=1}^m$

Unsupervised $\{x_i\}_{i=1}^m$

Target Function

$f : X \to Y$

Regression, Classification

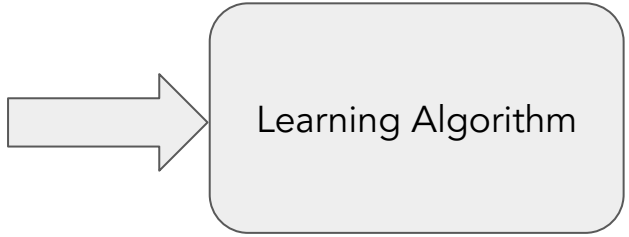
Density Estimation $p(x)$

Clustering $f : X \to Y$, $Y \in \{0, 1, \ldots, k\}$

Dimension Reduction $f : X \to Y$

$X \in \mathbb{R}^p, \quad Y \in \mathbb{R}^d$

# Supervised Learning vs. Unsupervised Learning




Training Data

Supervised $\{x^i, y^i\}_{i=1}^m$

Unsupervised $\{x_i\}_{i=1}^m$

Target Function

$f : X \to Y$

Regression, Classification

Density Estimation $p(x)$

Clustering $f : X \to Y$, $Y \in \{0, 1, \ldots, k\}$

Dimension Reduction $f : X \to Y$, $X \in \mathbb{R}^p, \quad Y \in \mathbb{R}^d$

## General ML Algorithm Pipeline

1. Build probabilistic models
2. Derive loss function:
   MLE vs. MAP
3. Select optimizer
   Necessary Condition vs. (Stochastic) GD

Q&A

Thanks for
Attending in the
whole semester