CS 333: Natural Language Processing

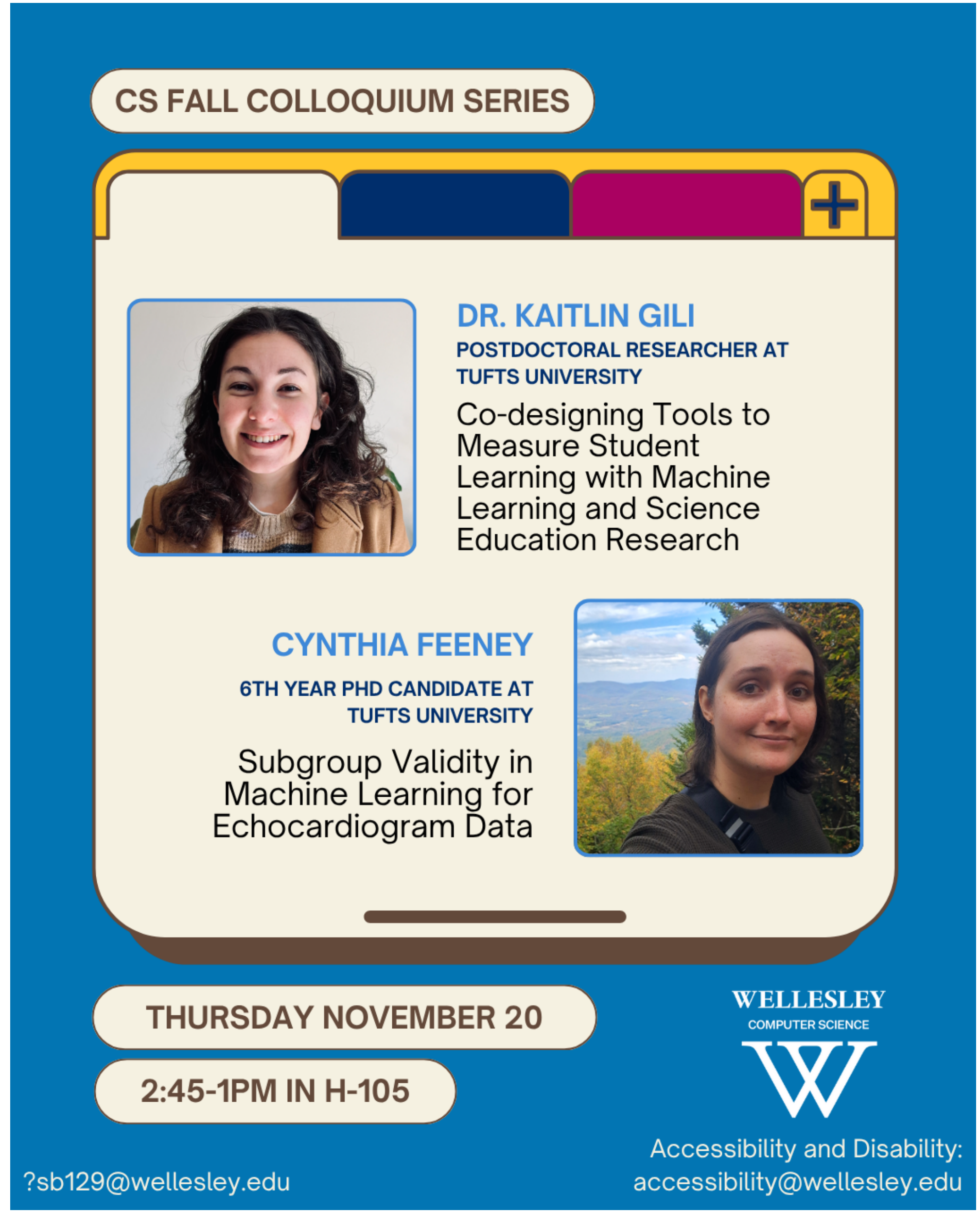
Fall 2025

Prof. Carolyn Anderson
Wellesley College

Reminders

- HW 8 will be released on today and due on Monday, 11/24
- My next help hours: Thursday 4-5
- Imposter Syndrome Survey (help out Sohie!): https://cs.wellesley.edu/~slee/imposterSyndrome

★ Prof. VanHattem is running a Coding Workshop on Thursday



The Deep Learning Pipeline

The Deep Learning Pipeline

Deep learning models can be run in two modes:

- **Training**: update a model's weights to fit new data. This is *supervised learning* because it requires input/output pairs (labeled data).

- **Inference**: run data through a model to make predictions. This requires only input data. It does not change the model weights.

Transfer Learning

Contemporary machine learning often involves multiple stages of training:

- **Pre-training**: train a large model that will be used by many downstream applications
  Called a foundation model in Bommasani et al. 2021

- **Fine-tuning**: adapting a pre-trained model to a new task or dataset by training it on new data, starting from existing weights.

  Meta models = public
  Qwen models

- **Prompt Engineering**: framing a task so that it can be solved by a pretrained language model.

-

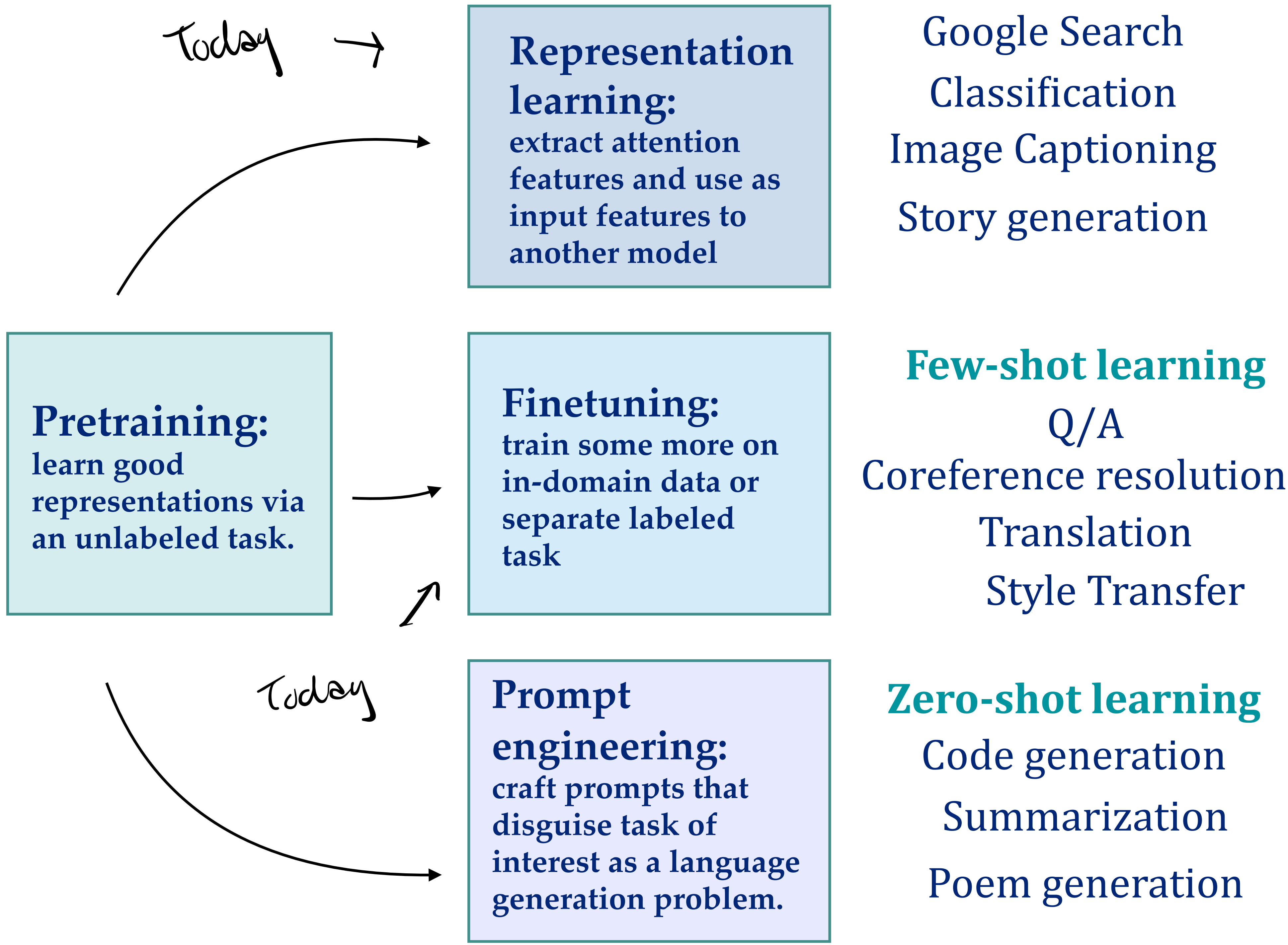
Transfer Learning

Contemporary machine learning models may also build upon other models by **freezing the weights of the original model** and taking some of its components as input.

For instance, the **weights of attention heads** may be re-used as embeddings to be fed in as input to a downstream model.

This is called **feature extraction**.

This is what we did in the ~~recipe~~ lyric classifier: we took attention weights from RoBERTa to use as features in our classifier!

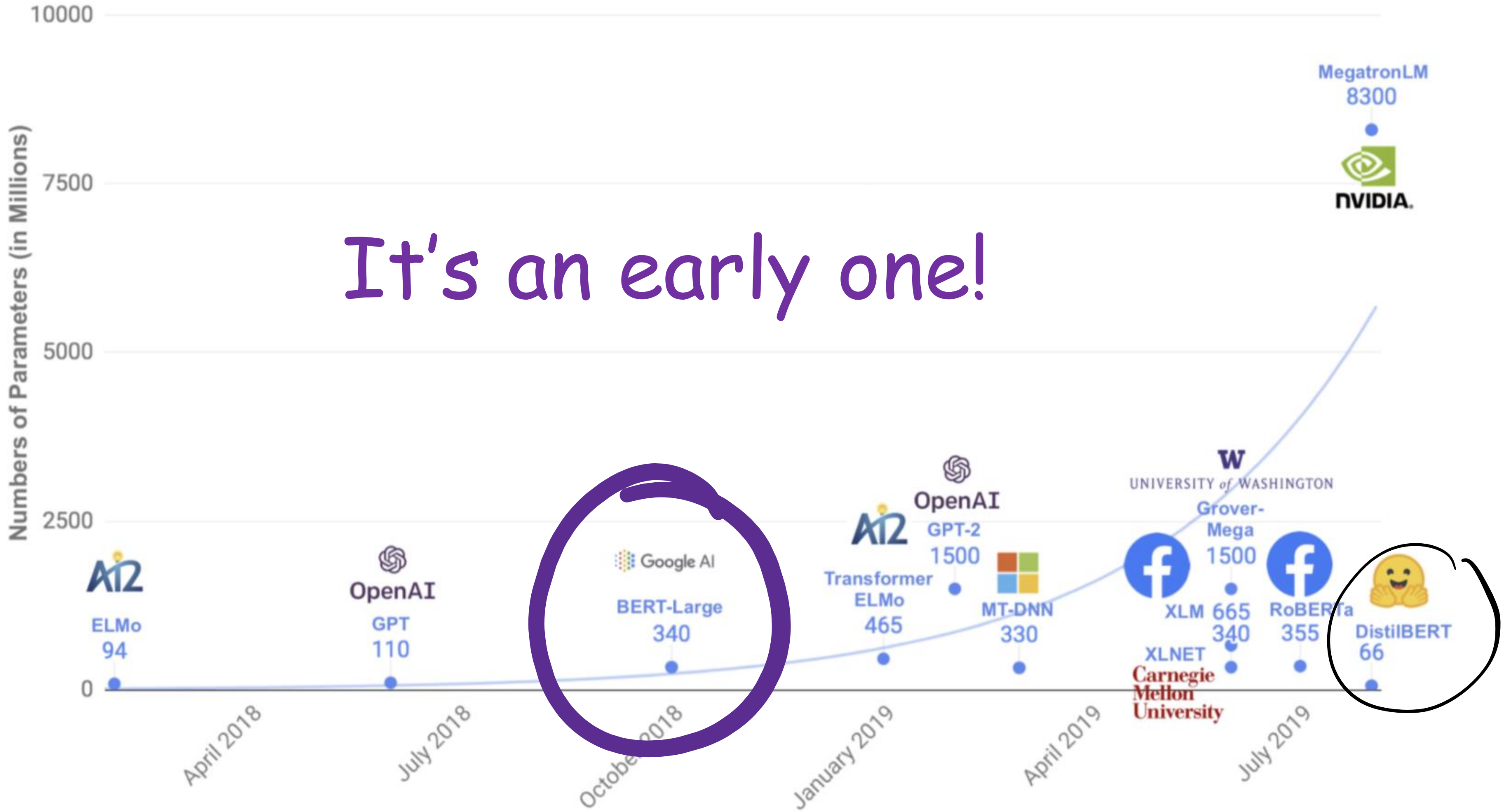

Today
Representation learning:
extract attention features and use as input features to another model
Google Search
Classification
Image Captioning
Story generation
Pretraining:
learn good representations via an unlabeled task.
Finetuning:
train some more on in-domain data or separate labeled task
Few-shot learning
Q/A
Coreference resolution
Translation
Style Transfer
Today
Prompt engineering:
craft prompts that disguise task of interest as a language generation problem.
Zero-shot learning
Code generation
Summarization
Poem generation

Representation Learning

BERT: **B**idirectional **E**ncoder **R**epresentations for **T**ransformers

Devlin et al. 2019

Why BERT?

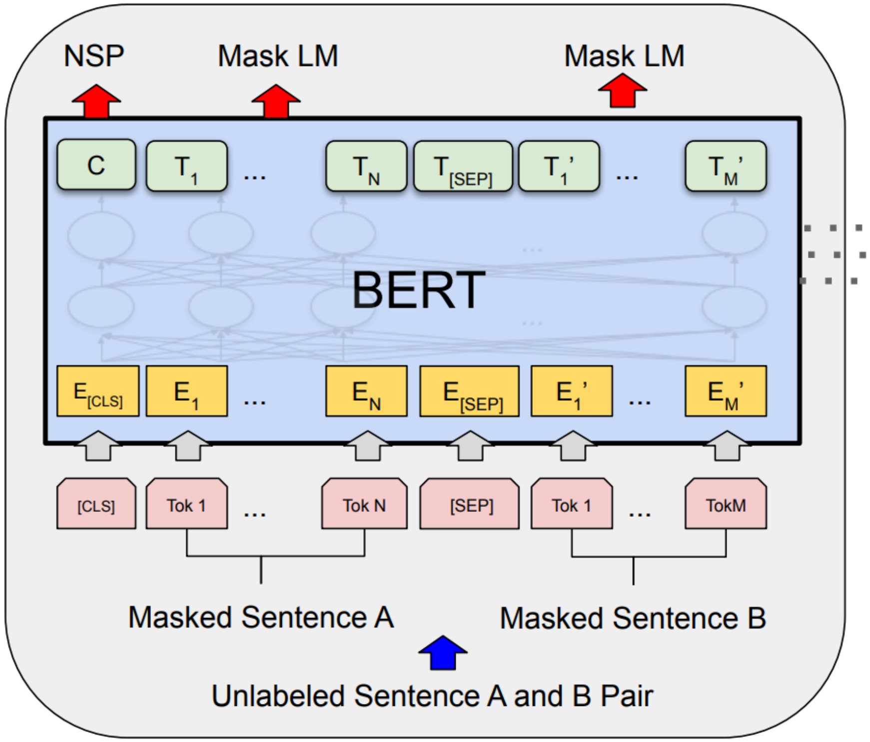


It's an early one!
10000
7500
5000
2500
0
Numbers of Parameters (in Millions)
ELMo
94
GPT
110
BERT-Large
340
Transformer
ELMo
465
GPT-2
1500
MT-DNN
330
XLM 665
XLNET
340
Grover-
Mega
1500
RoBERTa
355
DistilBERT
66
MegatronLM
8300
OpenAI
OpenAI
Google AI
UNIVERSITY of WASHINGTON
Carnegie Mellon University
NVIDIA
April 2018
July 2018
October 2018
January 2019
April 2019
July 2019

Pre-Training BERT Tasks

(1) Masked Language Model

(2) Next Sentence Prediction



Devlin et al. 2019

Masked Language Model Procedure

Example: `my dog is hairy`

- 80% of the time: Replace the word with the [MASK] token

  my dog is [MASK] and I ...

- 10% of the time: Replace the word with a random word

  my dog is apple

- 10% of the time: Keep the word unchanged

  my dog is hairy

Devlin et al. 2019

Masked Language Model Procedure

Example: `my dog is hairy`

- 80% of the time: Replace the word with the [MASK] token

  `my dog is [MASK]`

Bidirectional language modeling

- 10% of the time: Replace the word with a random word

  `my dog is apple`

- 10% of the time: Keep the word unchanged

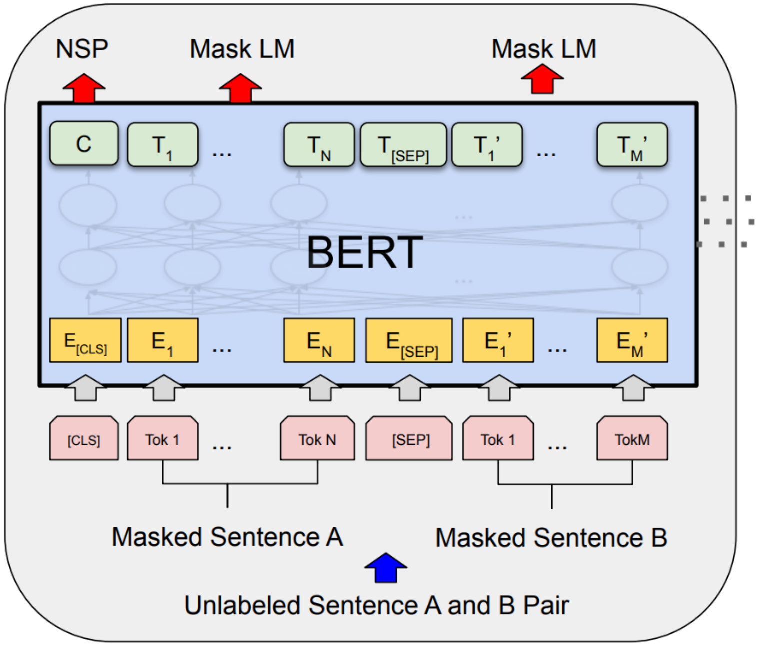
  `my dog is hairy`

Mitigate mismatch between pre-training & fine-tuning

Devlin et al. 2019

Pre-Training BERT: NSP

Idea: Predict whether sentence B follows sentence A using the final embedding of the `[CLS]` token



Devlin et al. 2019

BERT as Representation Learner

- By training on these tasks, which are self-supervised (labels for free), BERT learns good representations of word meaning.
- We can then extract word embeddings from BERT by taking the hidden state activations from the last layer (or from all layers).
- These embeddings can be used as input to another model or we could use them directly to evaluate textual similarity.

Example Application

Evaluating Computational Representations of Character: An Austen Character Similarity Benchmark

Funing Yang and **Carolyn Jane Anderson**
Wellesley College
Wellesley, MA
carolyn.anderson@wellesley.edu

Abstract

Several systems have been developed to extract information about characters to aid computa-
tional analysis o
pose character si
evaluation task f
AustenAlike, a l
similarities in Ja
mark draws on t
ilarity: a structu

James Morland from *Northanger Abbey*
Sibling to heroine and single 20-year-old male clergy with income of £400/year
Social Pairings: Charles Hayter, Edward Ferrars,

BookNLP

BookNLP is a natural language processing pipeline that scales to books and other long
including:

- Part-of-speech tagging
- Dependency parsing
- Entity recognition
- Character name clustering (e.g., "Tom", "Tom Sawyer", "Mr. Sawyer", "Thomas Saw
  and coreference resolution
- Quotation speaker identification
- Supersense tagging (e.g., "animal", "artifact", "body", "cognition", etc.)
- Event tagging

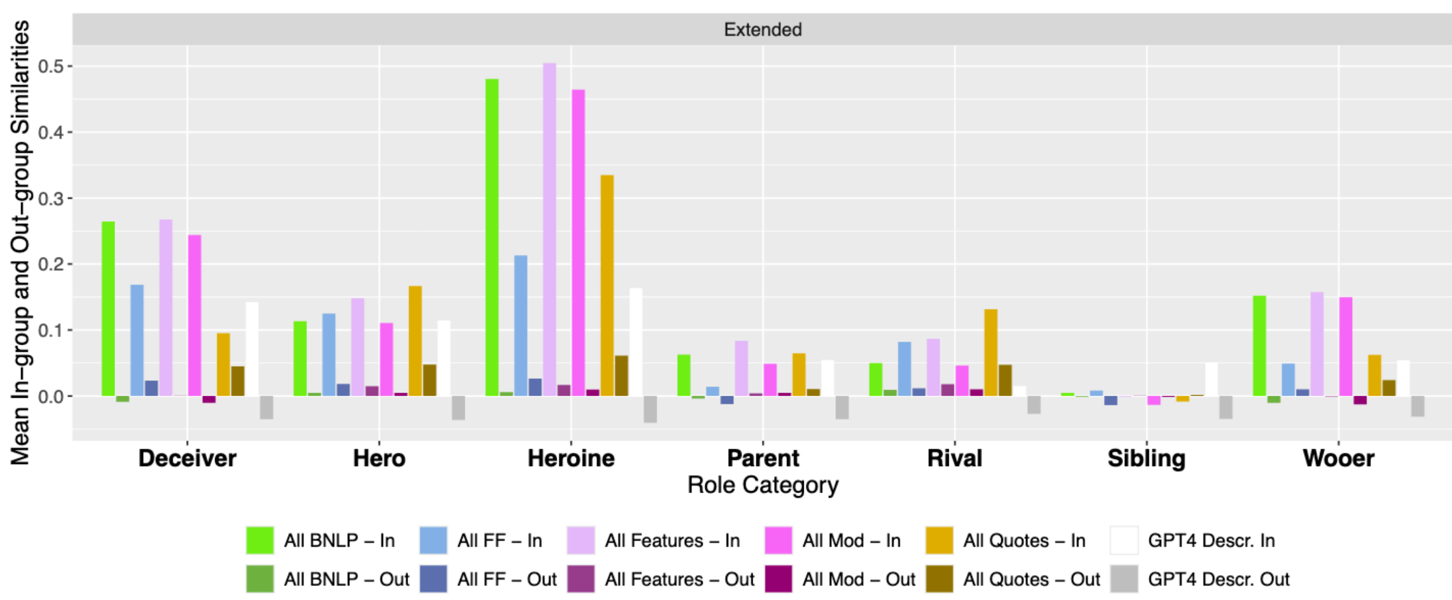


Figure 3: Narrative Role Benchmark: Mean cosine similarities between same-group characters and other characters, extended representations.

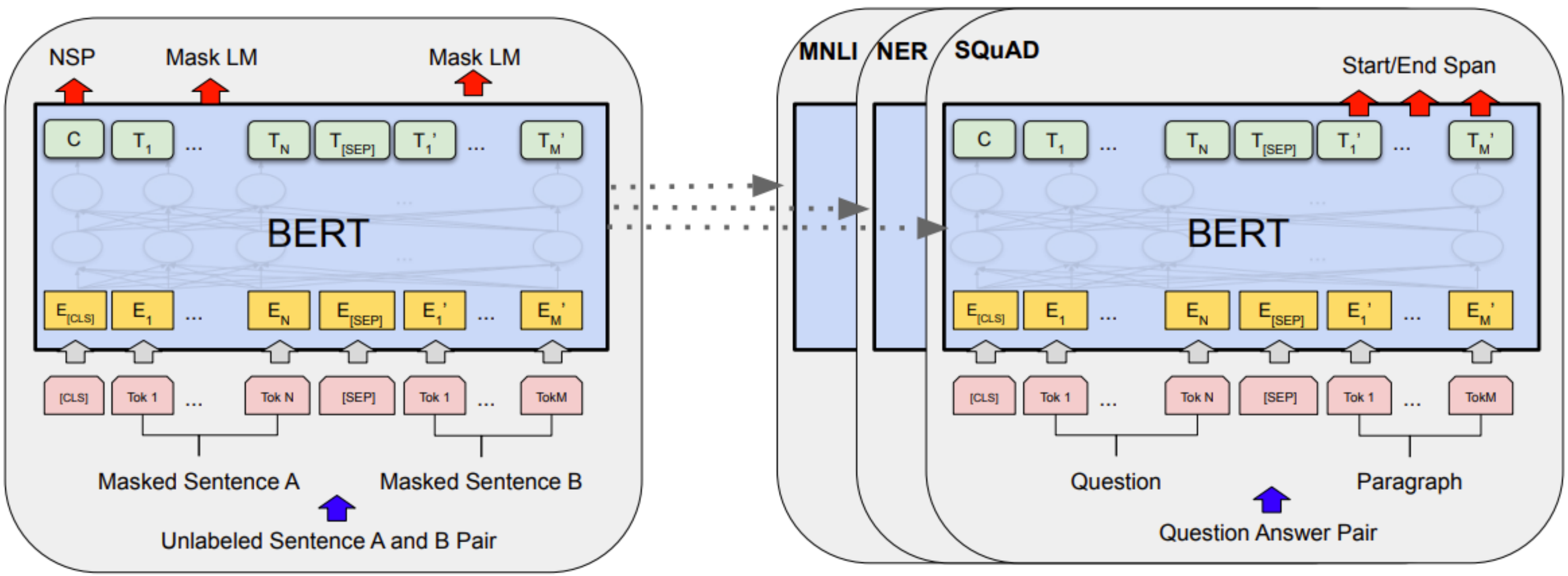
Fine-tuning

Fine-tuning = Further training

Simple Fine-tuning

Fine-tuning:
train some more on in-domain data or separate labeled task

Pre-Training vs. Fine-Tuning

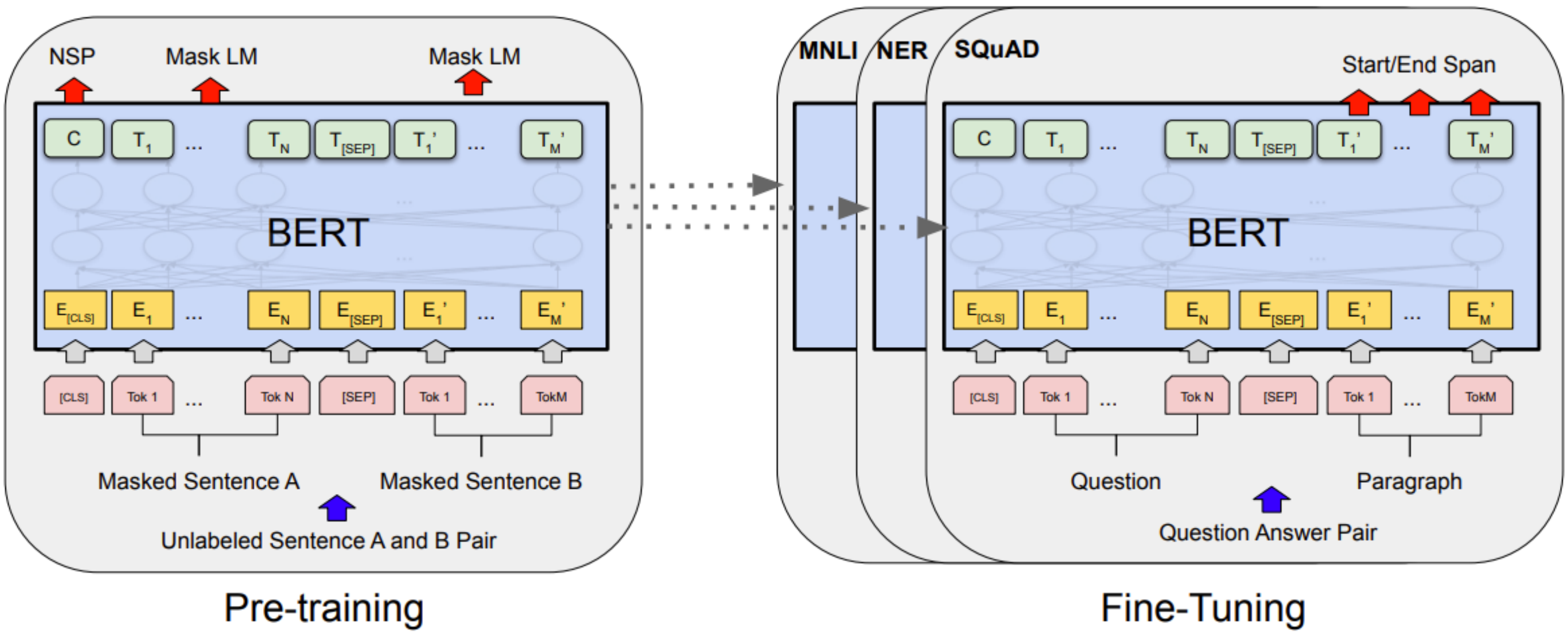


Pre-training

Fine-Tuning

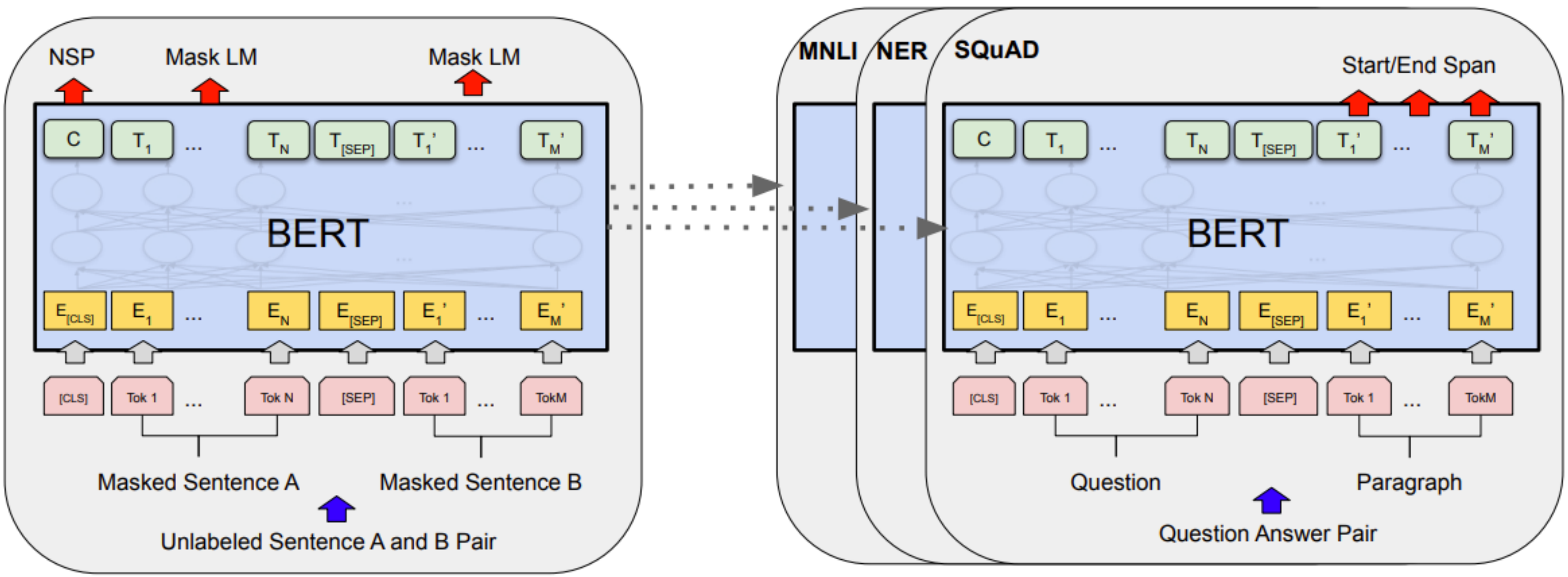
Devlin et al. 2019

Same internal architecture



Devlin et al. 2019

Different output layers & loss functions



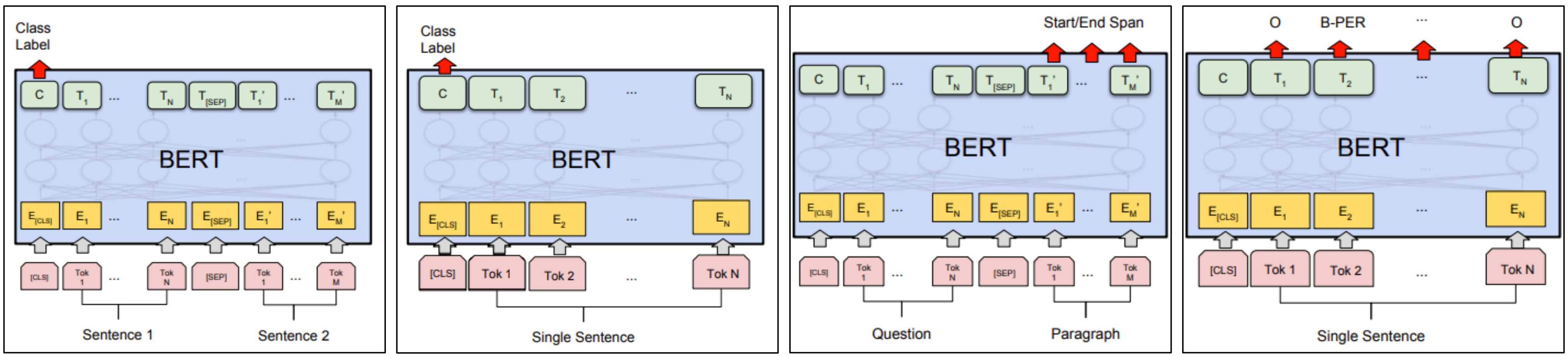
Pre-training Fine-Tuning

Devlin et al. 2019

Fine-Tuning

Use pre-trained **model parameters** for initialization

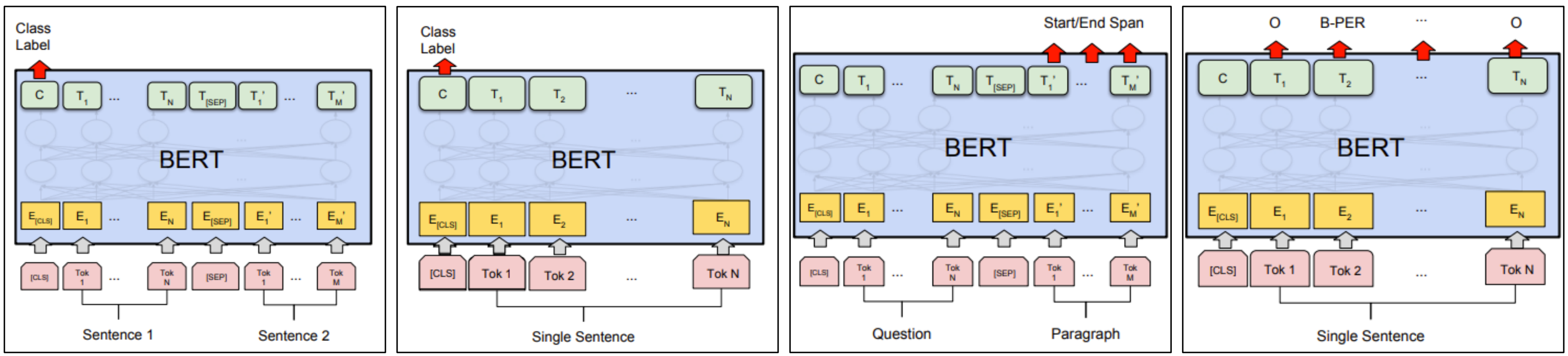
Change pre-training **output layers** of BERT to suit task



Devlin et al. 2019

Fine-Tuning

Sentence Pair Classification | Single Sentence Classification | Question Answering | Single Sentence Tagging



Devlin et al. 2019

Fancy Fine-tuning

Simple Fine-tuning Limitations

- Lack of computational resources: what if the model is **too big** for you to train?

Simple Fine-tuning Limitations

- Lack of computational resources: what if the model is **too big** for you to train?

You could fine-tune BERT for your final project, but you could not fine-tune Llama 8B…

Simple Fine-tuning Limitations

- Lack of computational resources: what if the model is **too big** for you to train?
- Lack of data: what if you can't find **labeled data** for your specific task?

Fine-tuning = Further training

Fine-tuning:
train some more on in-domain data or separate labeled task

Simple Fine-tuning

Just train on
on more data

Proxy Tuning

If model is too big, finetune a small model & use it to steer the large model.

Reinforcement Learning

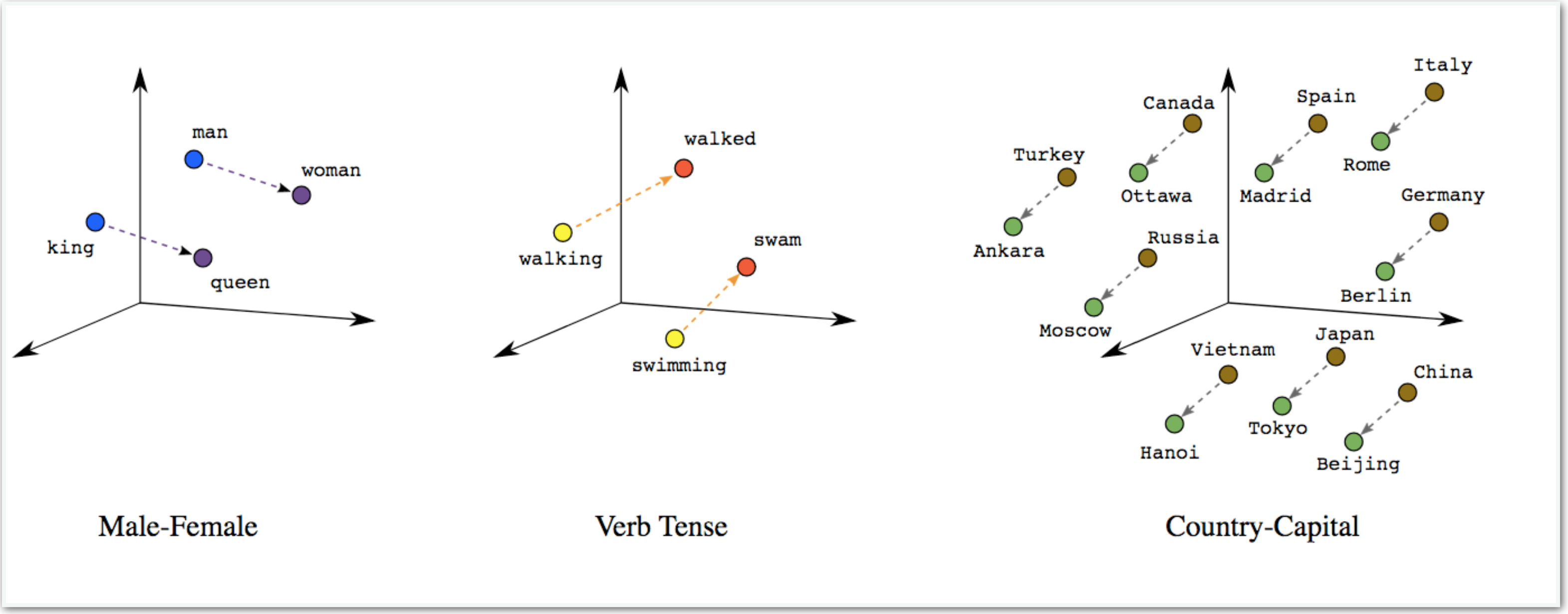
Tune a model
w/ a reward function

Proxy Tuning

Quick Refresher: Embedding Analogies

Back when we learned Word2Vec, we briefly talked about **embedding analogies**.

The idea is that there are reliable geometric relationship between embeddings that correspond to certain word relationships.



Proxy Tuning

Proxy tuning is a new technique useful when you have enough resources to fine-tune a small model, but not enough to fine-tune a larger model that you're interested in using.

Intuition:

Fine-tune the smaller model and use the **difference** in its pre/post fine-tuning predictions to alter the larger model's predictions.

Important! Models need the same tokenizer!

Proxy Tuning Paper

Published as a conference paper at COLM 2024

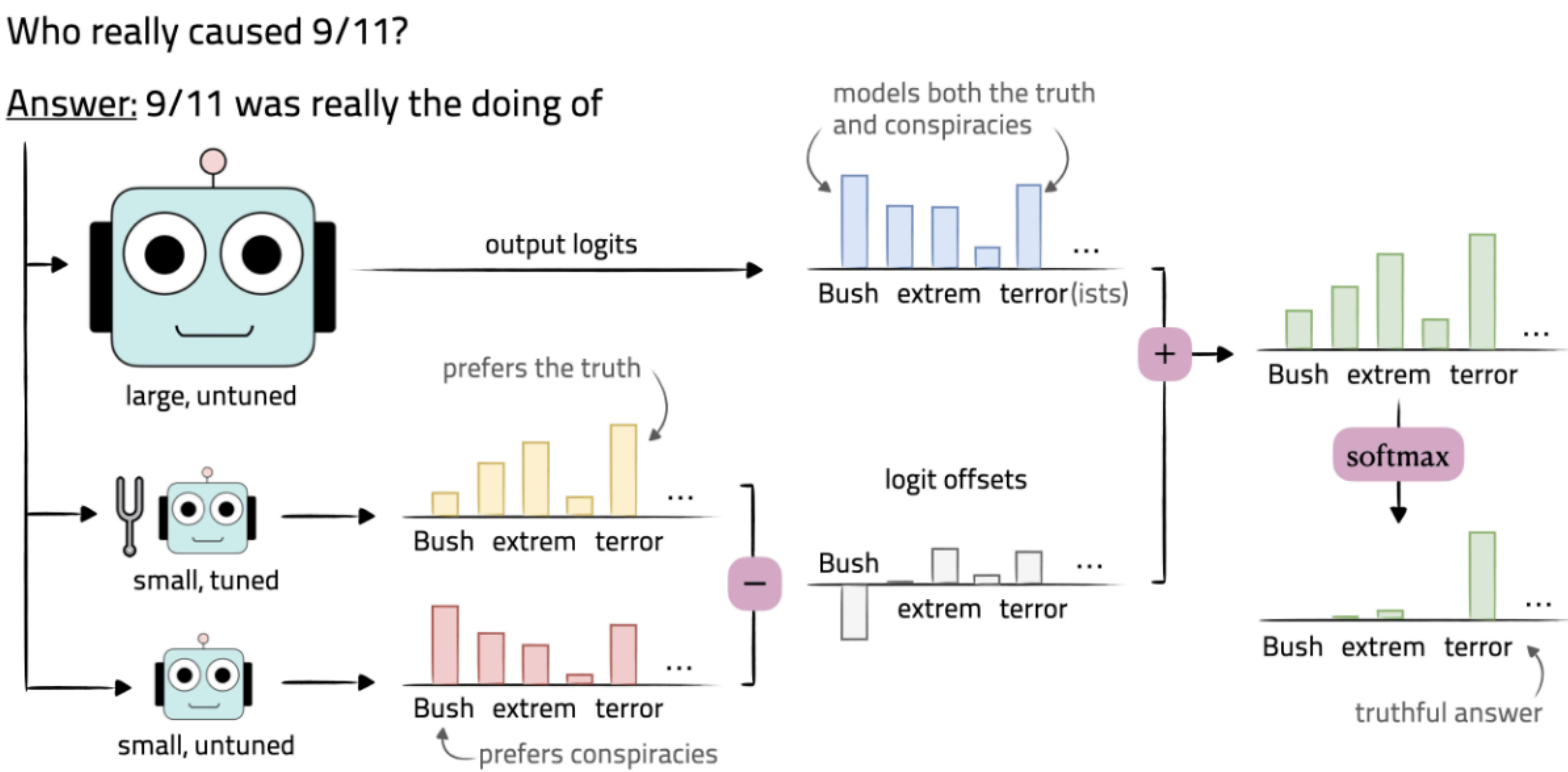
Tuning Language Models by Proxy

Alisa Liu[♡] **Xiaochuang Han**[♡] **Yizhong Wang**[♡♣] **Yulia Tsvetkov**[♡]
Yejin Choi[♡♣] **Noah A. Smith**[♡♣]
[♡]University of Washington [♣]Allen Institute for AI
`alisaliu@cs.washington.edu`

Abstract

Despite the general capabilities of large pretrained language models, they consistently benefit from further adaptation to better achieve desired behaviors. However, tuning these models has become increasingly resource-intensive, or impossible when model weights are private. We introduce **proxy-tuning**, a lightweight decoding-time algorithm that operates on top of black-box LMs to achieve the same end as direct tuning, but by accessing

Intuition: Steering a Larger Model



What We Need

- A small model that we can finetune
- A dataset to finetune on
- A large pretrained model that *gives us access to its logits*

Models

- Expert (M^+) : finetuned small model
- Antiexpert (M^-) : base small model
- Model (M) : base large model

How to Steer

At each timestep (each prediction site):

- Retrieve the logits from the expert: s_{M+}
- Retrieve the logits from the antiexpert: s_{M-}
- Take the difference between the two: $s_{M+} - s_{M-}$
- Retrieve the logits from the model: s_M
- Apply the difference to the model logits:

$$s'_M = s_M + (s_{M+} - s_{M-})$$

- Softmax to obtain the steered prediction

Intuition: Steering a Larger Model

Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Answer: $1__

	6 8 4 0 …
Expert:	[1 5 0.5 -1 …]
Anti-expert:	[3 3 5.5 -1 …]
Large:	[3 2 0.1 0.5 …]
Proxy-Tuned:	[1 4 -4.9 0.5 …]

} [-2 2 -5 0]

Answer: $18

Instruction Proxy-Tuning Results

Model	AlpacaFarm (↑) Win rate	GSM (↑) Acc.	ToxiGen (↓) % Toxic	TruthfulQA (↑) MC Acc.	TruthfulQA (↑) % Info + True
7B					
Directly tuned	82.5	23.0	0.0	55.9	81.3
13B					
Base (untuned)	2.1	6.6	70.4	38.6	49.1
Proxy-tuned	83.4	26.4	0.1	57.4	**82.0**
Directly tuned	**87.3**	**32.4**	**0.0**	**61.6**	80.4
70B					
Base (untuned)	3.7	9.6	67.4	42.3	53.9
Proxy-tuned	88.0	32.0	**0.0**	59.2	**85.1**
Directly tuned	**90.4**	**51.8**	**0.0**	**68.3**	79.6

Table 2: **Results for instruction-tuning.** For each model size, **Base** refers to the pretrained LLAMA2 model, **Directly tuned** refers to LLAMA2-CHAT, and the **Proxy-tuned** model always uses LLAMA2-7B-CHAT as the expert and LLAMA2-7B as the anti-expert. Overall, proxy-tuning dramatically improves performance over the base model, on average closing 91.1% and 88.1% of the gap with the corresponding CHAT model at 13B and 70B size, respectively. It also outperforms the small expert alone in all scenarios except a 0.1% difference in ToxiGen.

Task Proxy-tuning Results

Model	CodexEval	DS-1000
7B		
Directly tuned	68.9	53.6
13B		
Base (untuned)	33.7	26.2
Proxy-tuned	65.7	42.8
Directly tuned	**78.6**	**56.9**
70B		
Base (untuned)	62.0	43.9
Proxy-tuned	70.7	50.6
Directly-tuned	**89.2**	**67.6**

Table 4: **Results for code adaptation. Directly tuned** refers to CODELLAMA-PYTHON. The **proxy-tuned** model uses CODELLAMA-7B-PYTHON as the expert, and LLAMA2-7B as the anti-expert. The metric is pass@10 (↑).

Model	TriviaQA	GSM
7B		
Directly tuned	55.8	40.6
13B		
Base (untuned)	36.8	6.6
Proxy-tuned	55.9	43.9
Directly tuned	**59.5**	**51.0**
70B		
Base (untuned)	45.2	9.6
Proxy-tuned	62.7	53.9
Directly tuned	**63.1**	**67.9**

Table 5: **Results for task-specific tuning. Directly tuned** refers to a task expert obtained by finetuning LLAMA2 on either TriviaQA or GSM. The **proxy-tuned** model uses the 7B task model as the expert and LLAMA2-7B as the anti-expert.

HW 8

In HW 8, we will replicate the GSM results from the proxy-tuning paper, but with smaller models. We will use Llama 3.2 1B as our expert/antiexpert, and Llama 3.1 8B as our larger model.

Three parts:

1. Evaluation: how well do the base models perform on GSM?
2. Finetuning: fine-tune an (even smaller) model on GSM
3. Proxy-tuning: steer Llama 3.1 8B with a finetuned Llama 3.2 1B model.

HW 8

Three parts:

1. Evaluation: how well do the base models perform on GSM? *Can be done in Colab without GPU access.*

2. Finetuning: fine-tune an (even smaller) model on GSM. *Requires GPU access. Do this on the NSF Delta platform.*

3. Proxy-tuning: steer Llama 3.1 8B with a finetuned Llama 3.2 1B model. *Can be done in Colab without GPU access.*

Fine-tuning = Further training

Fine-tuning:
train some more on in-domain data or separate labeled task

Simple Fine-tuning

Just train on more (labeled) data

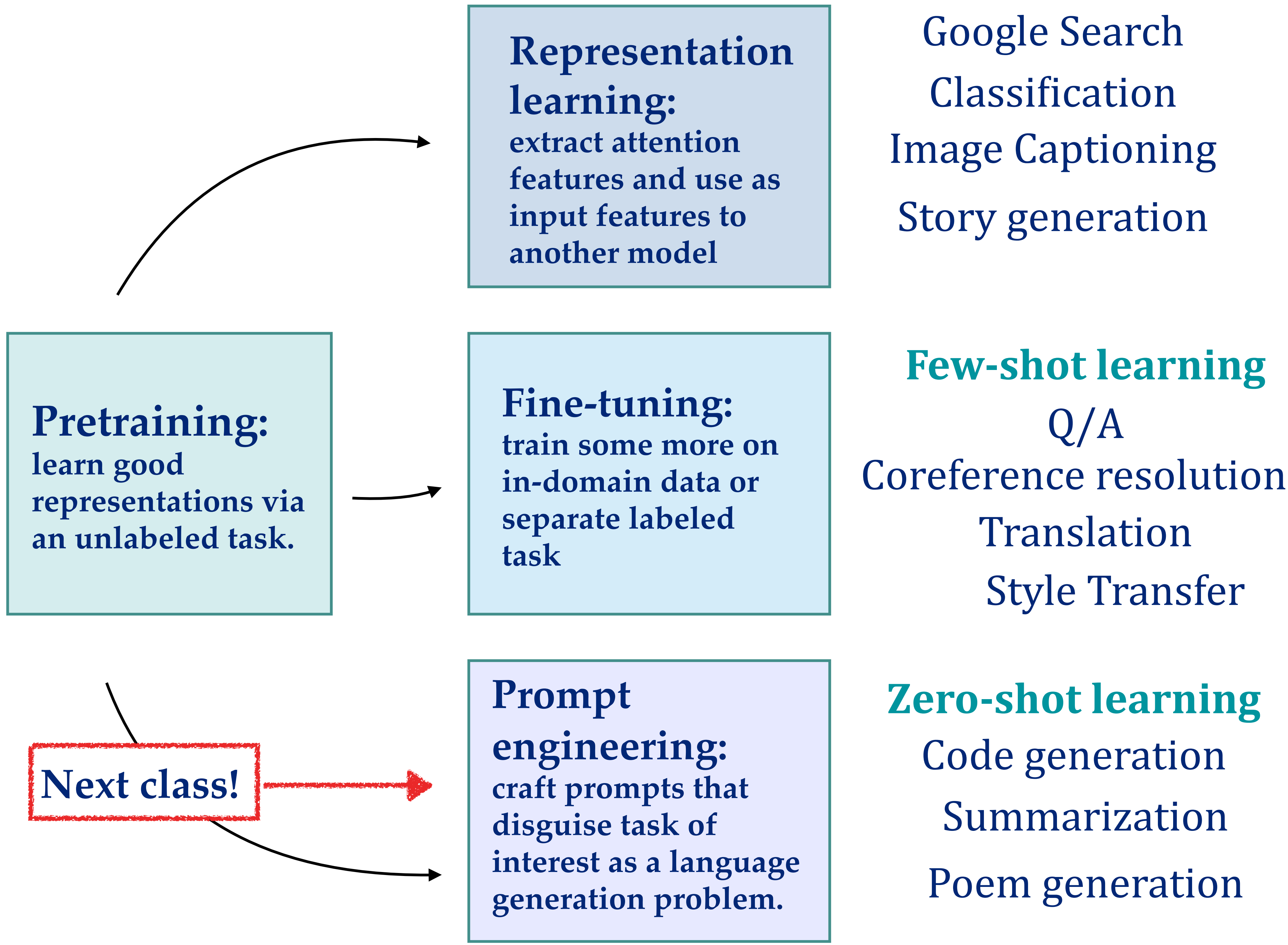
Proxy Tuning

Fine-tune a smaller model and use it steer a larger model

Reinforcement Learning

Tune the model with a reward function

Next class! → Reinforcement Learning


Pretraining:
learn good representations via an unlabeled task.
Representation learning:
extract attention features and use as input features to another model
Google Search
Classification
Image Captioning
Story generation
Fine-tuning:
train some more on in-domain data or separate labeled task
Few-shot learning
Q/A
Coreference resolution
Translation
Style Transfer
Next class!
Prompt engineering:
craft prompts that disguise task of interest as a language generation problem.
Zero-shot learning
Code generation
Summarization
Poem generation