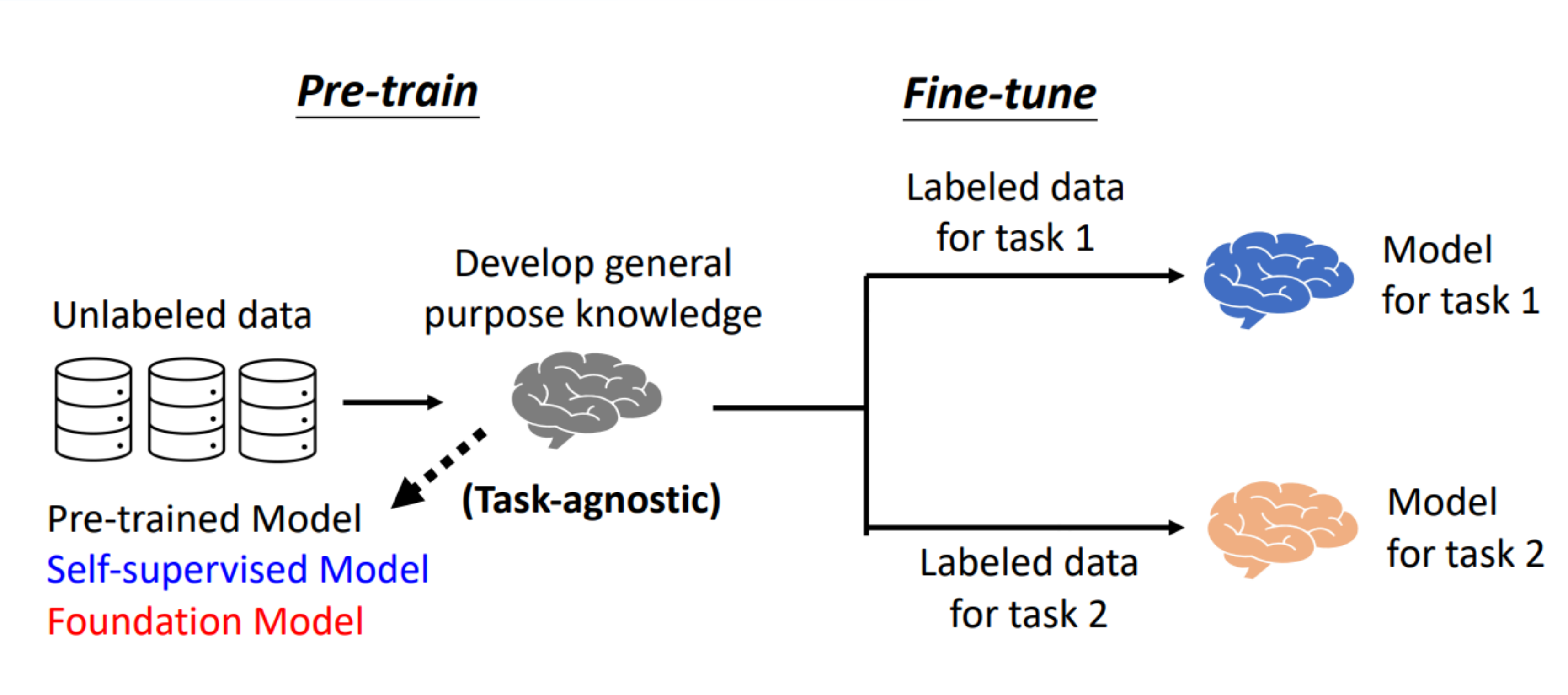
Summer School Week4

洪畯宸

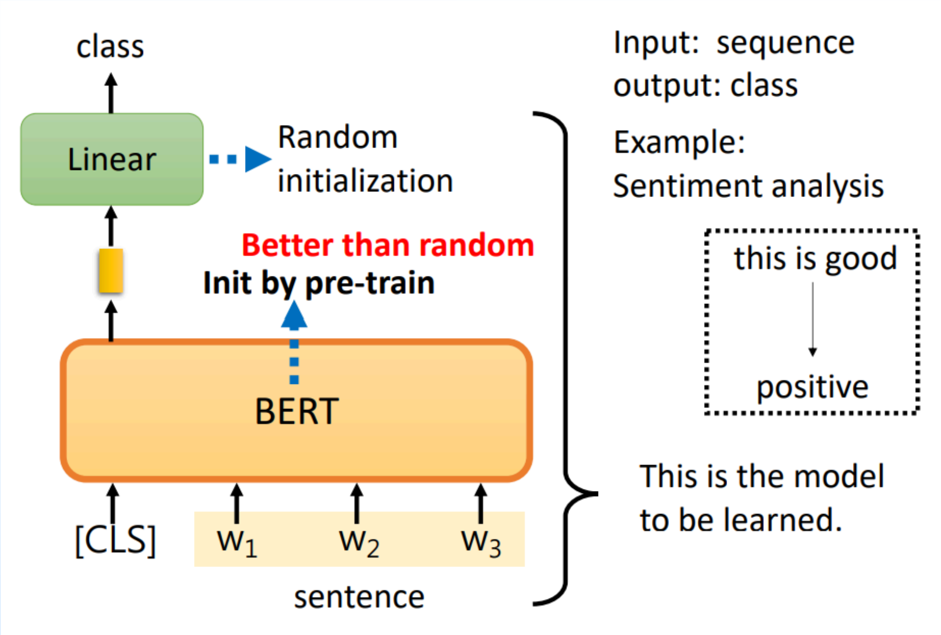
Slide here!

Introduction & How do PLM works

Framework of Pre-training



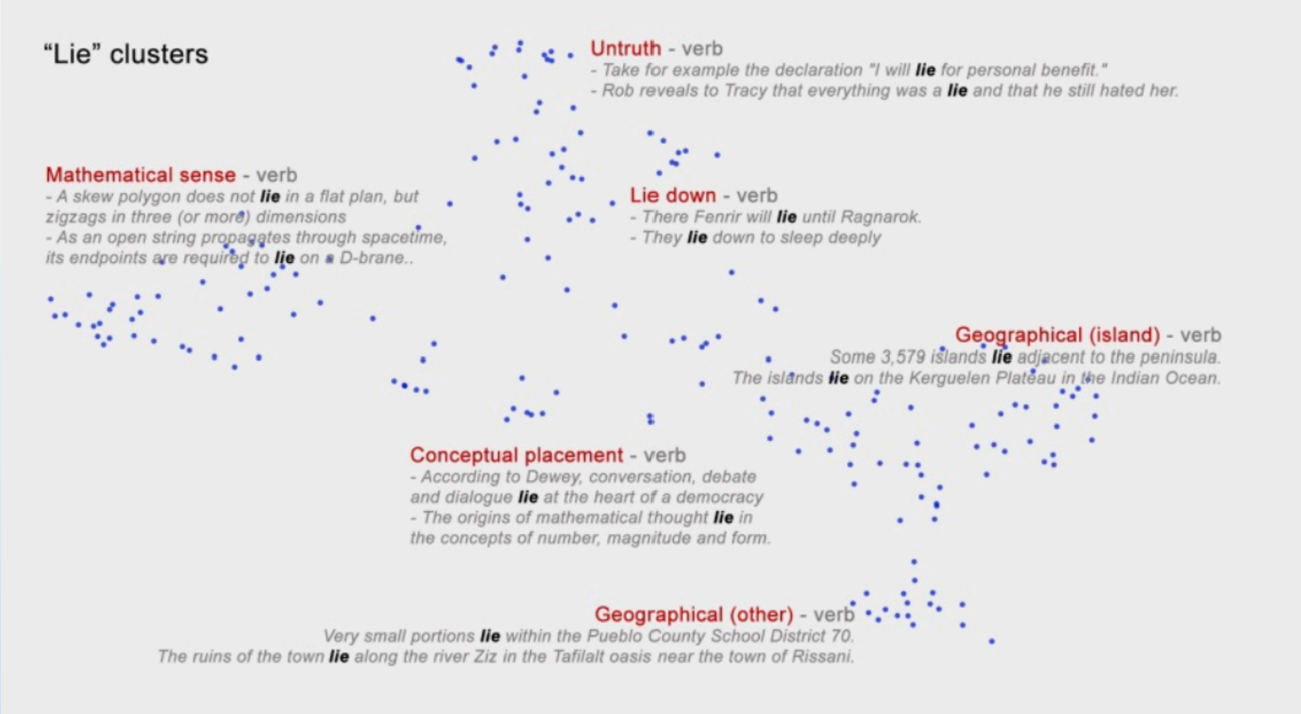
Pre-training for NLP



- Use the benefit form pre-traing parameters so we don’t have to traing the whole model

Contextualized Word Representations

- The tokens with similar meaning have similar embedding.



BERTology - What does each layer learn?

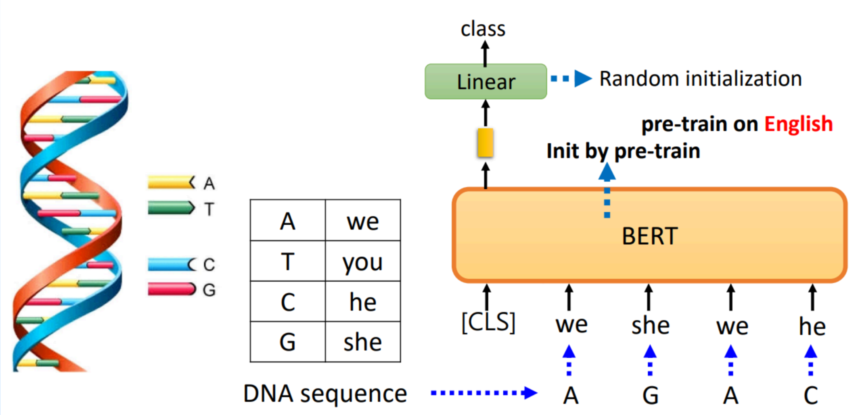
- Higher classifier accuracy does not always mean encoding more information.
- BERT understand a sentance order: Surface => Sytavtic => Semantic
  - There are no a clear line of each step, they may be mixed.

Analyzing what BERT learned during training

- BERT Embryology: When does BERT know POS tagging, syntactic parsing, semantics?
- When Do You Need Billions of Words of Pretraining Data?

Cross-discipline Capability

- Use human language pretrained model to do not human languages tasks, like DNA classification

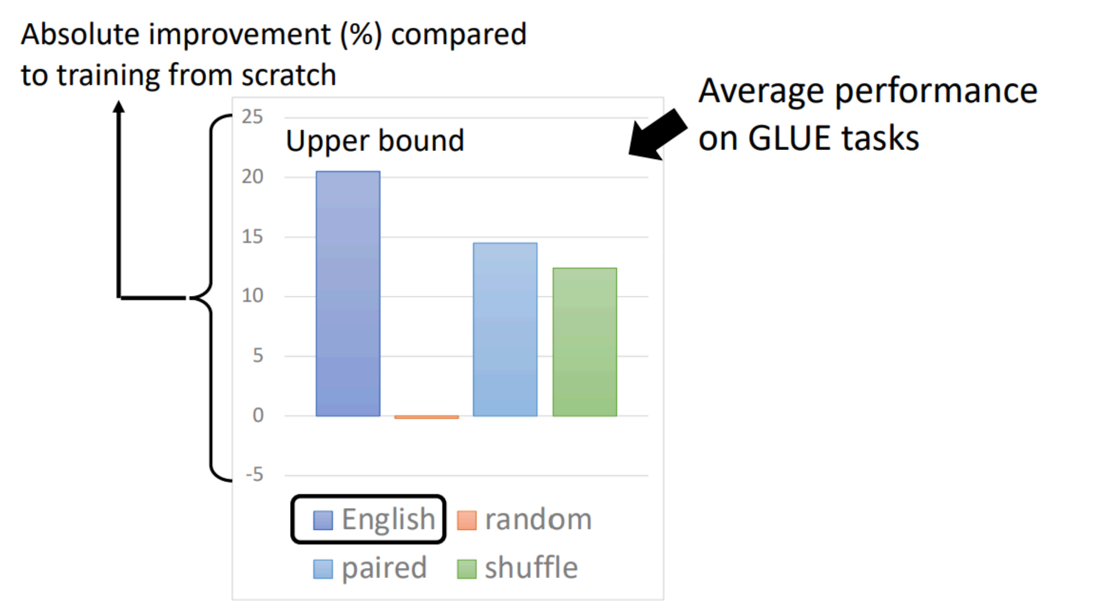


	Protein			DNA				Music
	localization	stability	fluorescence	H3	H4	H3K9ac	Splice	composer
specific	69.0	76.0	63.0	87.3	87.3	79.1	94.1	-
BERT	64.8	74.5	63.7	83.0	86.2	78.3	97.5	55.2
re-emb	63.3	75.4	37.3	78.5	83.7	76.3	95.6	55.2
rand	58.6	65.8	27.5	75.6	66.5	72.8	95	36

- The pretrained models learn some general skills for the classification

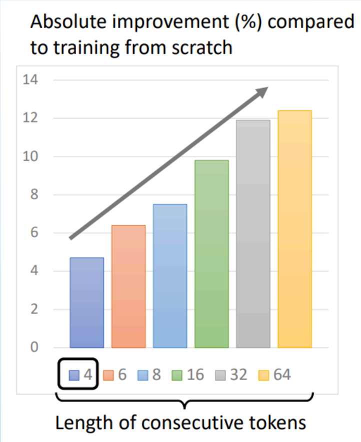
Pre-training on Artificial Data

- By generating artificial data with different rules, we can know what are the key factors for the success of pre-training.



Pre-training on Artificial Data

- English: The upper bound
- Random: Very low, which means data plays the role
- Paired: Structured data is critical for learning useful skills for NLP
- Shuffle: Long-range reading may be the key to the success of a pretrained model?
- Learning to read a long-range in a sequence is crucial.



How to use PLMs: Contrastive Learning for PLMs

Contrastive Learning

- Similar inputs have similar representations -> Positive Pairs
- Dissimilar inputs have dissimilar representations. -> Negative Pairs
- When apply on NLP:
    - eg: `How is the [MASK] today ?`
    - Positive: non-contextualized representation of “weather”
    - Negative: non-contextualized representation of all the other words in the vocabulary
- **Sentence-level task!**: We have infinite possible sentences; not possible to enumerate all the sentences in the world.
    - Good to apply contrastive learning for sentence-level representations.

Why we need sentence-level representations?

- Provide as a **backbone** that can be useful on a variety of downstream sentence-level tasks
- Good generalization ability on tasks without much training data
  - e.g. even **linear probing** can achieve good performance
- Efficient sentence-level **clustering** or **semantic search** by inner products
- Measure similarities among sentence pairs
- Unsupervised methods are more desirable in order to be applied to languages beyond English

How to obtain sentence-level representations from BERTs?

- It cannot be trivially obtained from token-level representations
- Average pooling performs even worse than avg. GloVe embeddings
- **Representation degeneration**: the learned embeddings occupy a narrow cone in the vector space
  - Limits the expressiveness of the vector space

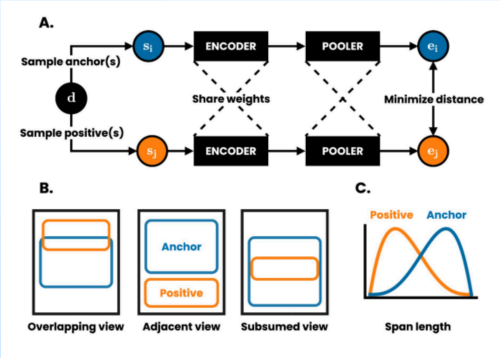
Post-processing Methods

- BERT-flow: map to a smooth isotropic semantic space
- BERT-whitening

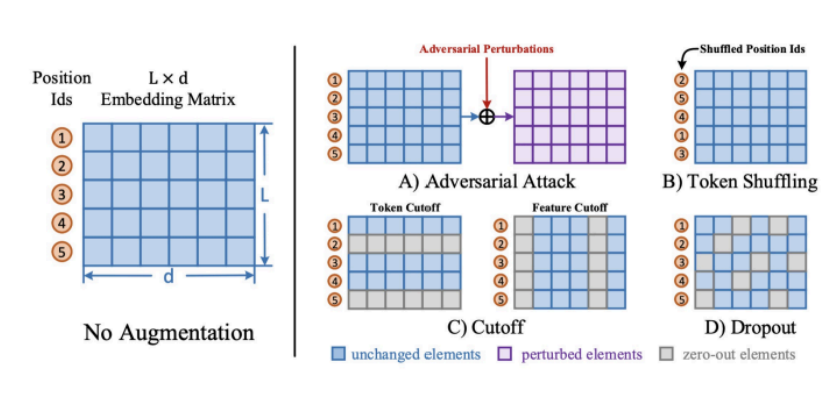
	STS-B	STS-12	STS-13	STS-14	STS-15	STS-16	SICK-R
			Published in (Reimers and Gurevych, 2019)				
Avg. GloVe embeddings	58.02	55.14	70.66	59.73	68.25	63.66	53.76
Avg. BERT embeddings	46.35	38.78	57.98	57.98	63.15	61.06	58.40
BERT CLS-vector	16.50	20.16	30.01	20.09	36.88	38.03	42.63
			Published in (Li et al., 2020)				
$\text{BERT}_{\text{base}}$-first-last-avg	59.04	57.84	61.95	62.48	70.95	69.81	63.75
$\text{BERT}_{\text{base}}$-flow (NLI)	58.56	59.54	64.69	64.66	72.92	71.84	**65.44**
$\text{BERT}_{\text{base}}$-flow (target)	70.72	63.48	72.14	68.42	73.77	**75.37**	63.11
			Our implementation				
$\text{BERT}_{\text{base}}$-first-last-avg	59.04	57.86	61.97	62.49	70.96	69.76	63.75
$\text{BERT}_{\text{base}}$-whitening (NLI)	68.19(↑)	61.69(↑)	65.70(↑)	66.02(↑)	**75.11**(↑)	73.11(↑)	63.6(↓)
$\text{BERT}_{\text{base}}$-whitening-256 (NLI)	67.51(↑)	61.46(↑)	66.71(↑)	66.17(↑)	74.82(↑)	72.10(↑)	64.9(↓)
$\text{BERT}_{\text{base}}$-whitening (target)	71.34(↑)	63.62(↑)	73.02(↑)	**69.23**(↑)	74.52(↑)	72.15(↓)	60.6(↓)
$\text{BERT}_{\text{base}}$-whitening-256 (target)	**71.43**(↑)	**63.89**(↑)	**73.76**(↑)	69.08(↑)	74.59(↑)	74.40(↓)	62.2(↓)

Contrastive Learning Methods: Designed Positives

- **DeCLUTR**
  - **Positive**: Overlapping/adjacent spans from the same document
  - **Negative**: hard negatives from same docs, easy negatives from different docs



- **ConSERT**
  - All the possible augmentations on token embedding space



Contrastive Learning Methods: Generating Positives

- Continuations generated by GPT-2 XL

Task: Write two sentences that mean the same thing.

Sentence 1: "A man is playing a flute."

Sentence 2: "He's playing a flute."

Task: Write two sentences that are somewhat similar.

Sentence 1: "A man is playing a flute."

Sentence 2: "A woman has been playing the violin."

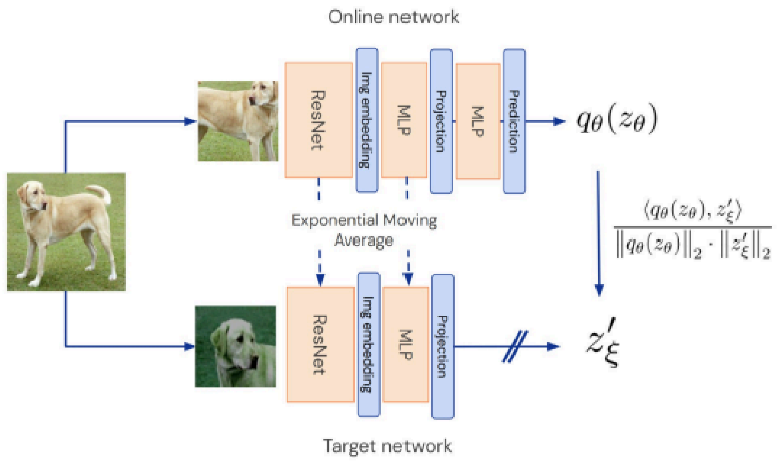
Task: Write two sentences that are on completely different topics.

Sentence 1: "A man is playing a flute."

Sentence 2: "A woman is walking down the street."

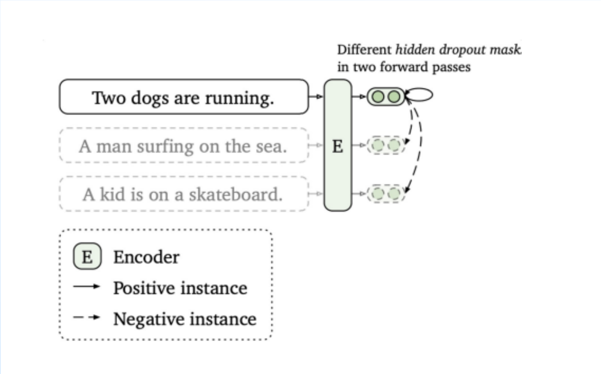
Contrastive Learning Methods: Bootstrapping Methods

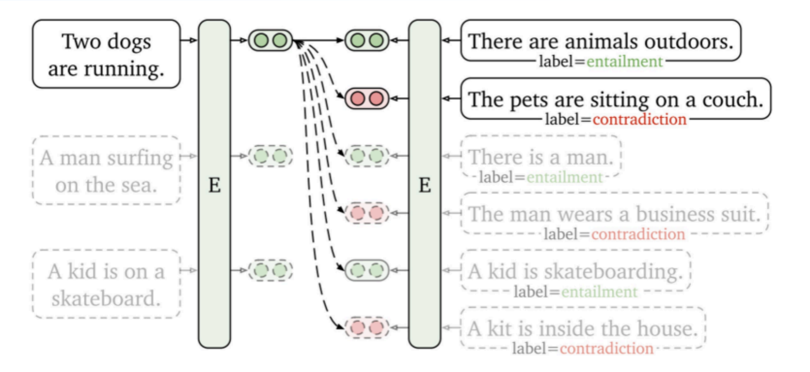
- BYOL
  - Not contrastive learning
  - Only positive pairs, no negatives pairs
  - Use a **moving average target network** to prevent mode collapsing



Contrastive Learning Methods: Dropout Augmentations

- SimCSE: Using different dropout masks (in Transformer layers) as augmentation
  - Model architecture is the same

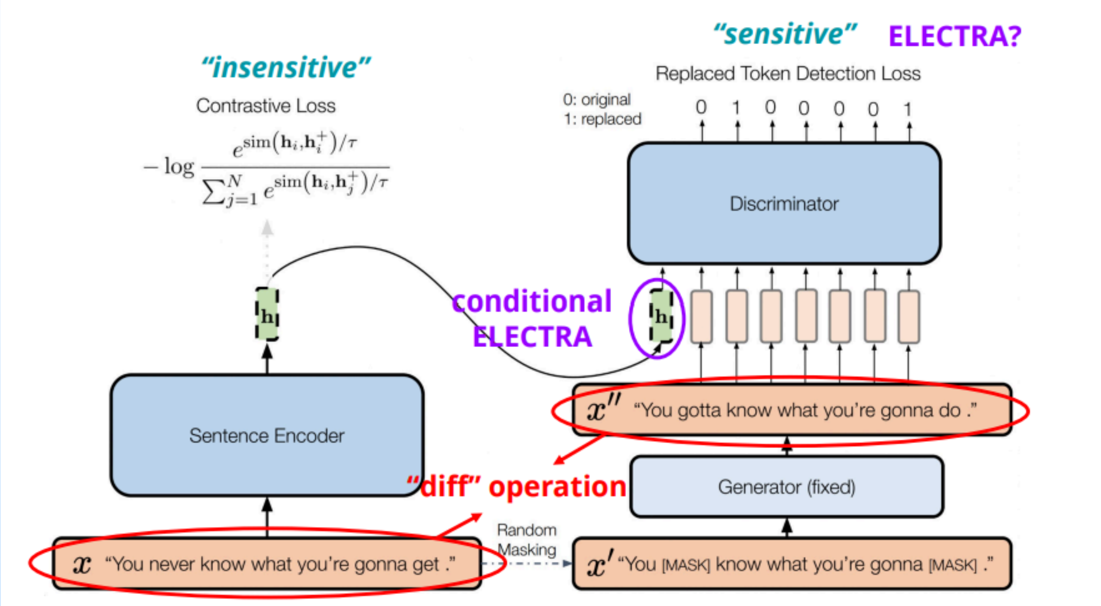




- mSimCSE: Contrastive learning on **only English** data with multilingual models (mBERT, XLM-R) can align all other other languages **without any parallel data**.

Contrastive Learning Methods: Equivariant Contrastive Learning

- DiffCSE



Contrastive Learning Methods: Prompting

- PromptBERT
  - Design/search good prompt templates to better extract sentence embeddings from BERT without fine-tuning
  - Further fine-tuning with contrastive loss:
    - Using sentence vectors produced by two different templates as a positive pair

Contrastive Learning Methods: Ranking-based Methods

- RankEncoder
  - Refine the vector space of existing models like SimCSE, PromptBERT
  - Leverage ranking information from the **whole corpus**
  - Train a new encoder to match the cosine similarity of rank vectors
  - RankEncoder can be aware of the fine-grain interaction between the **similar sentences** in the corpus
  - Closing the gap between unsupervised and supervised sentence representations

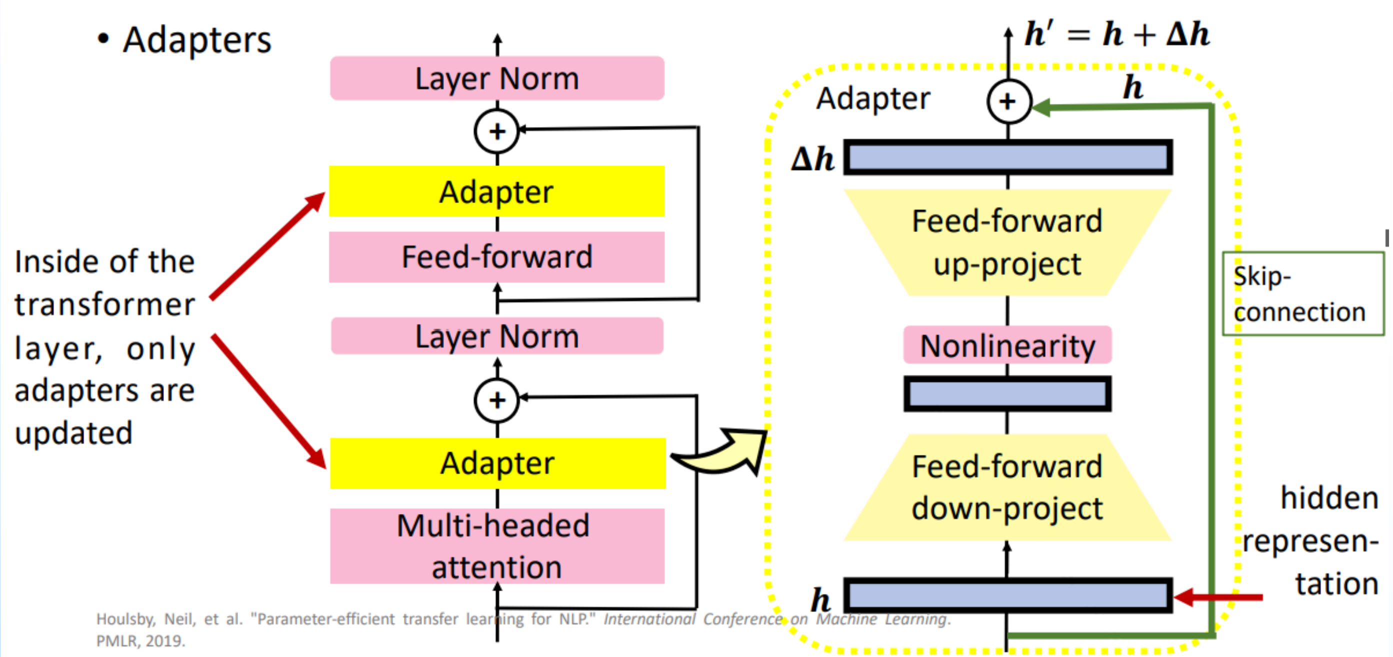
How to use PLMs: Parameter-efficient fine-tuning

Why Parameter-efficient fine-tuning

- Problem: PLMs are gigantic (in terms of numbers of parameters, model size, and the storage needed to store the model)
- Solution: Reduce the number of parameters by parameter-efficient fine-tuning
- Comparison:
  - Standard Fine-tuning: Directly modify hidden representations => Gigantic
  - Parameter-Efficient Fine-tuning: Adds on hidden representations, can choose where to add. The original parmeter will stay still

PLM: Adapter

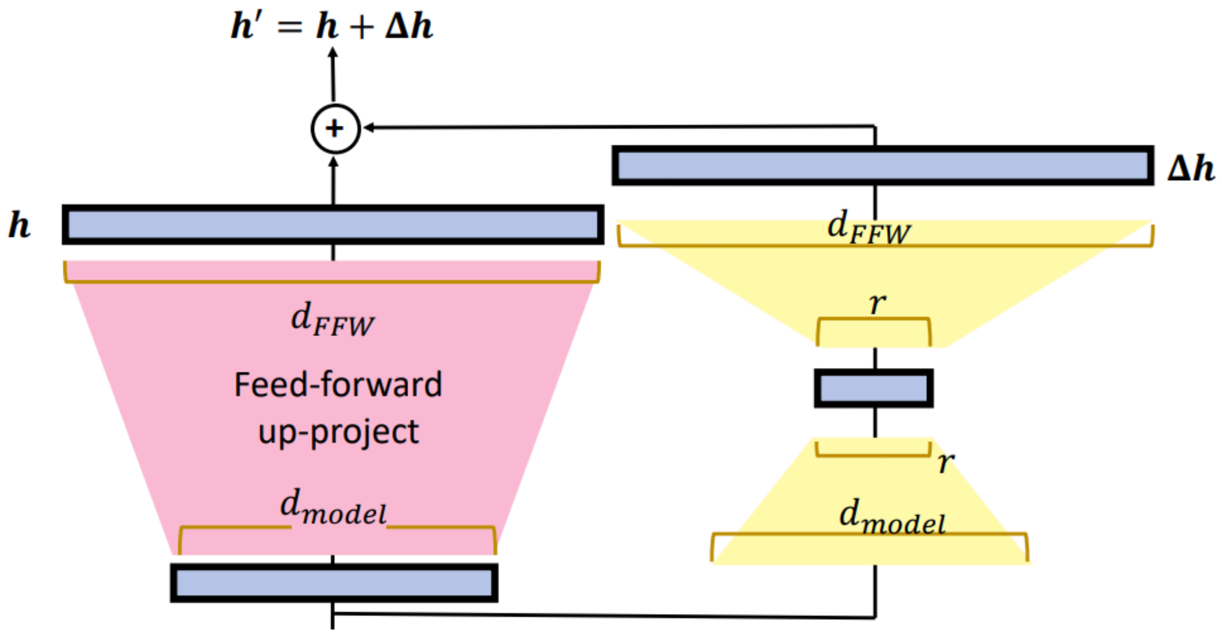
- Add after Multi-head Attention and Feed-Forward Layer



Houlsby, Neil, et al. "Parameter-efficient transfer learning for NLP." *International Conference on Machine Learning*. PMLR, 2019.

PLM: LORA

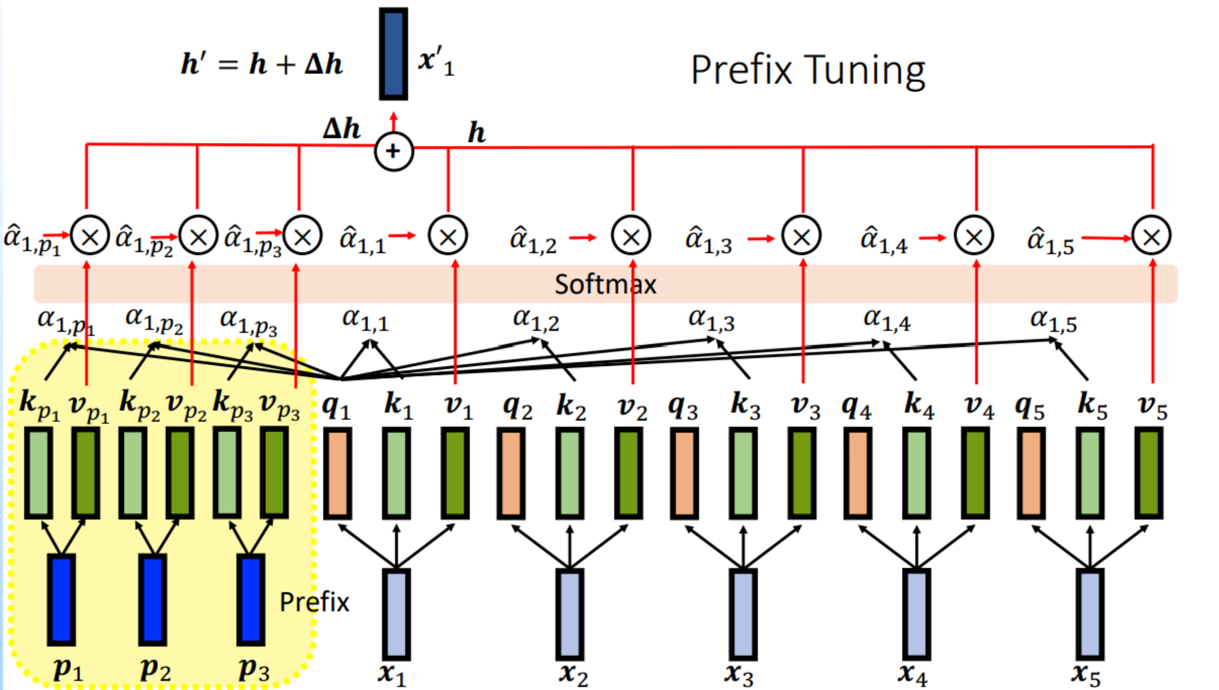
- Add on Feed-Forward Layer



Hu, Edward J., et al. "LoRA: Low-Rank Adaptation of Large Language Models." *International Conference on Learning Representations*. 2021.

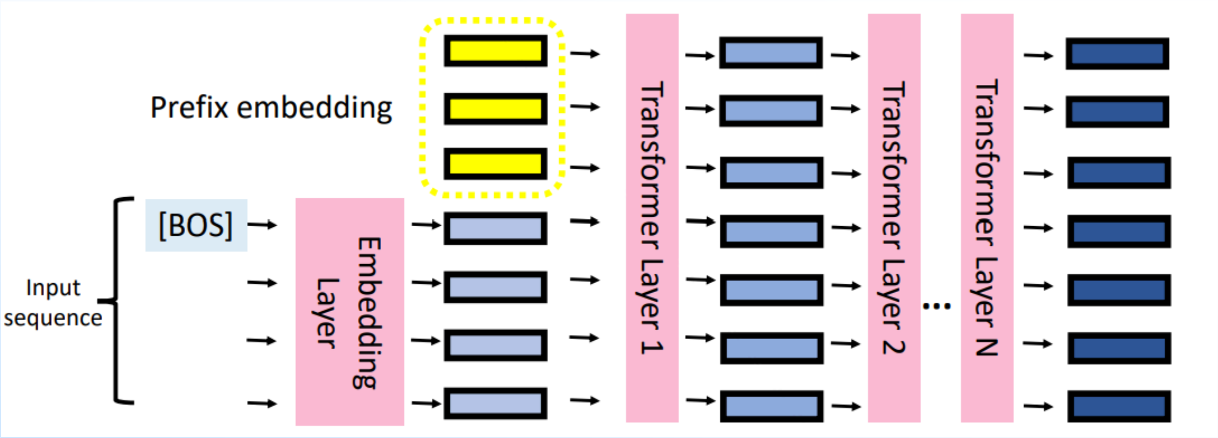
PLM: Prefix Tuning

- Insert trainable prefix in each layer



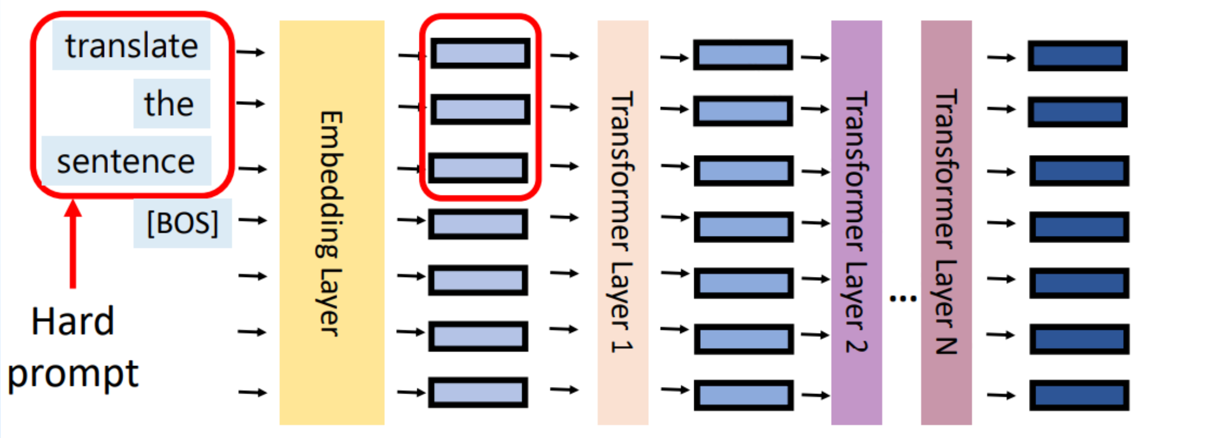
PLM: Prompt Tuning

- Soft Prompting: Prepend the prefix embedding at the input layer



PLM: Prompt Tuning

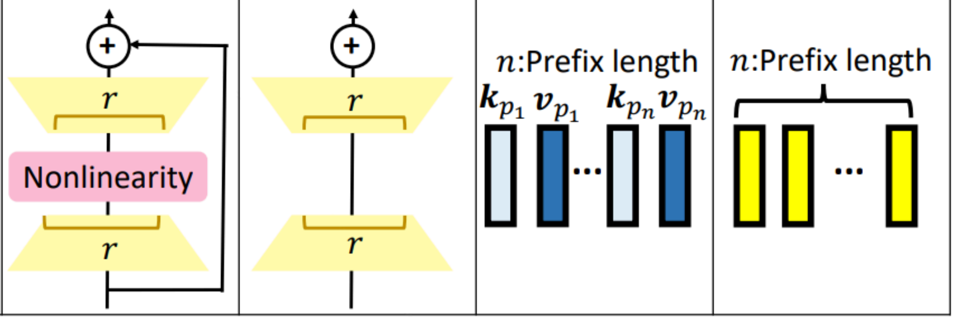
- Hard Prompting: add words in the input sentence



PLM: Benefits

- Drastically decreases the task-specific parameters
- Less easier to overfit on training data; better out-of-domain performance
- Fewer parameters to fine-tune, making them good candidates when training with small dataset

	Adapter	LoRA	Prefix Tuning	Soft Prompt
Task-specific parameters*	$\Theta(d_{model}rL)$	$\Theta(d_{model}rL)$	$\Theta(d_{model}nL)$	$\Theta(d_{model}n)$
Percent Trainable	<5%	<0.1%	<0.1%	<0.05%
Trainable parameters Illustration				



How do PLMs work: Using PLMs with different amounts of data

Intermediate-task fine-tuning

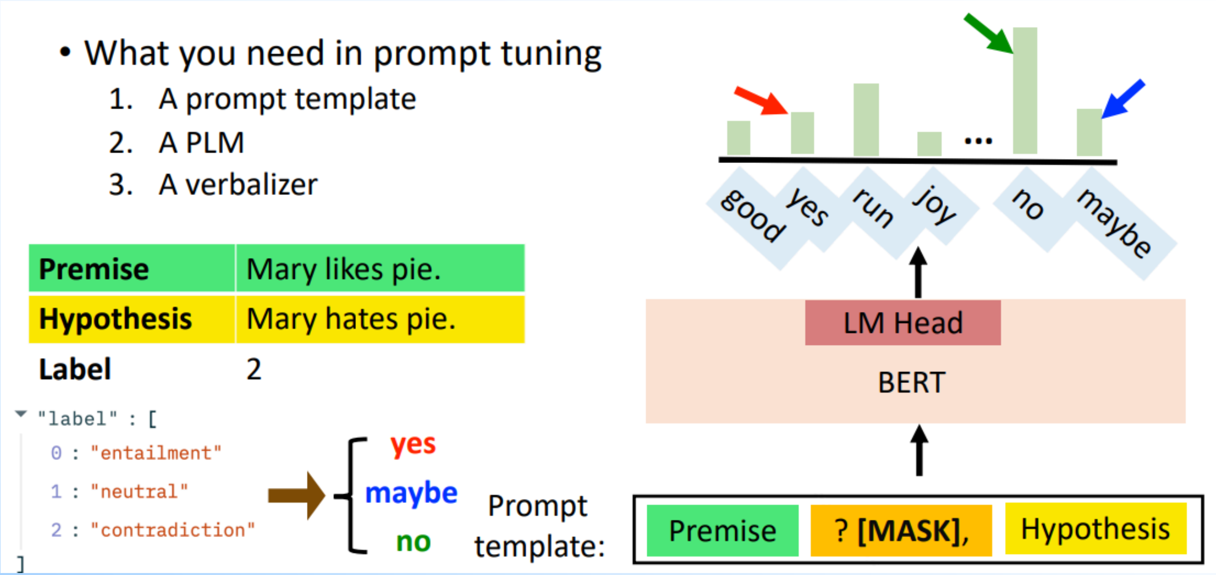
- Transfer the knowledge from a model finetuned on other tasks
- Conclusions:
    - Same type of tasks is the most beneficial
    - Even when the intermediate-task or the target task has limited data
    - **Soft Prompt Transfer (SPoT)**:
        - When fine-tuning with soft prompt tuning, we only need to transfer the prompt embedding instead of the whole model
        - The soft prompt of a task can be used as the task embedding of that task

Multi-task fine-tuning

- Fine-tune the PLM using the auxiliary task datasets and the target task dataset simultaneously

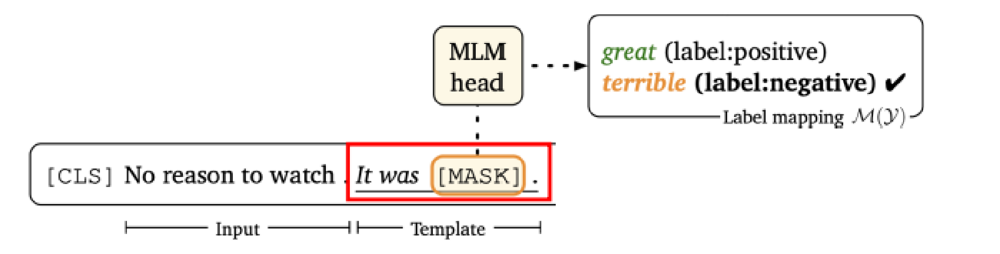
Prompt tuning for few-shot learning

- Pprompt template: convert data points into a natural language prompt
- PLM: perform language modeling
- Verbalizer: A mapping between the label and the vocabulary



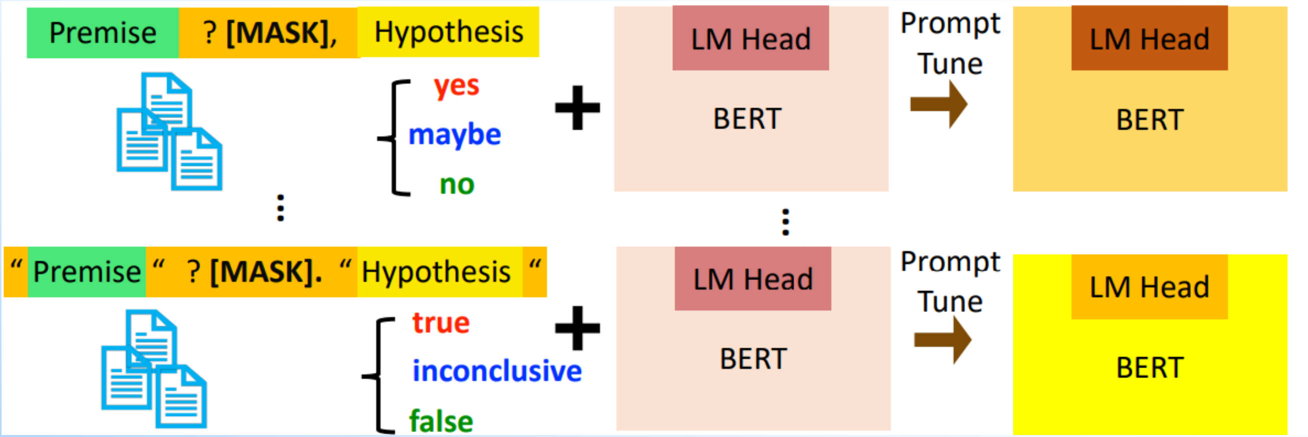
Prompt tuning for few-shot learning

- Prompt tuning has better performance under data scarcity because
  - It incorporates human knowledge
  - It introduces no new parameters
- How to select the verbalizer?
  1. **Manual design**: require task-specific knowledge
  2. **Prototypical verbalizer**: use learnable prototype vectors to represent a class, instead of using the words in the vocabulary
- Improve: LM-BFF



Semi-supervised learning with PLMs

- Use the labeled data to train a good model and use that model to label the unlabeled data
- Pattern-Exploiting Training (PET)
  - Use different prompts and verbalizer to prompt-tune different PLMs on the labeled dataset
  - Predict the unlabeled dataset and combine the predictions from different models
  - Use a PLM with classifier head to train on the soft-labeled data set

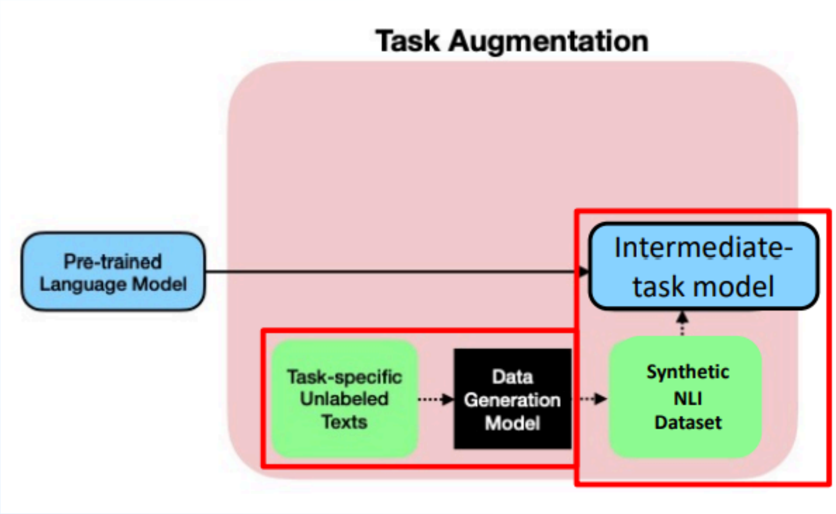


Semi-supervised learning with PLMs

- STraTa: Self-Training with Task Augmentation
  - Use unlabeled data to generate an NLI dataset, and finetuned on the NLI dataset as the intermediate task to obtain the base model

Steps

1. Train an NLI data generator using another labeled NLI dataset using a generative language model
2. Use the trained data generator to generate NLI dataset using the in-domain unlabeled data
3. Use the generated in-domain NLI dataset to fine-tune an NLI model. The finetuned model is used to initialize the teacher model and student model in self-training



Zero-shot learning

- During pre-training, the training datasets implicitly contains a mixture of different tasks
- Encoder-decoder model pretrained using MLM is the best

Evaluating LLM-based Applications

Why evaluation?

- Models are constantly updating
- LLMs makes tons of mistakes
- New prompt looks better in a few examples, but may not be better in general
- It helps:
  - Validation that model avoids common failure modes
  - Common language of go/not go decisions
  - Roadmap for improvements to model performance

What makes evaluation difficult?

- Trained on the internet => drift
- Qualitative => hard to measure success
- Diversity of behaviors

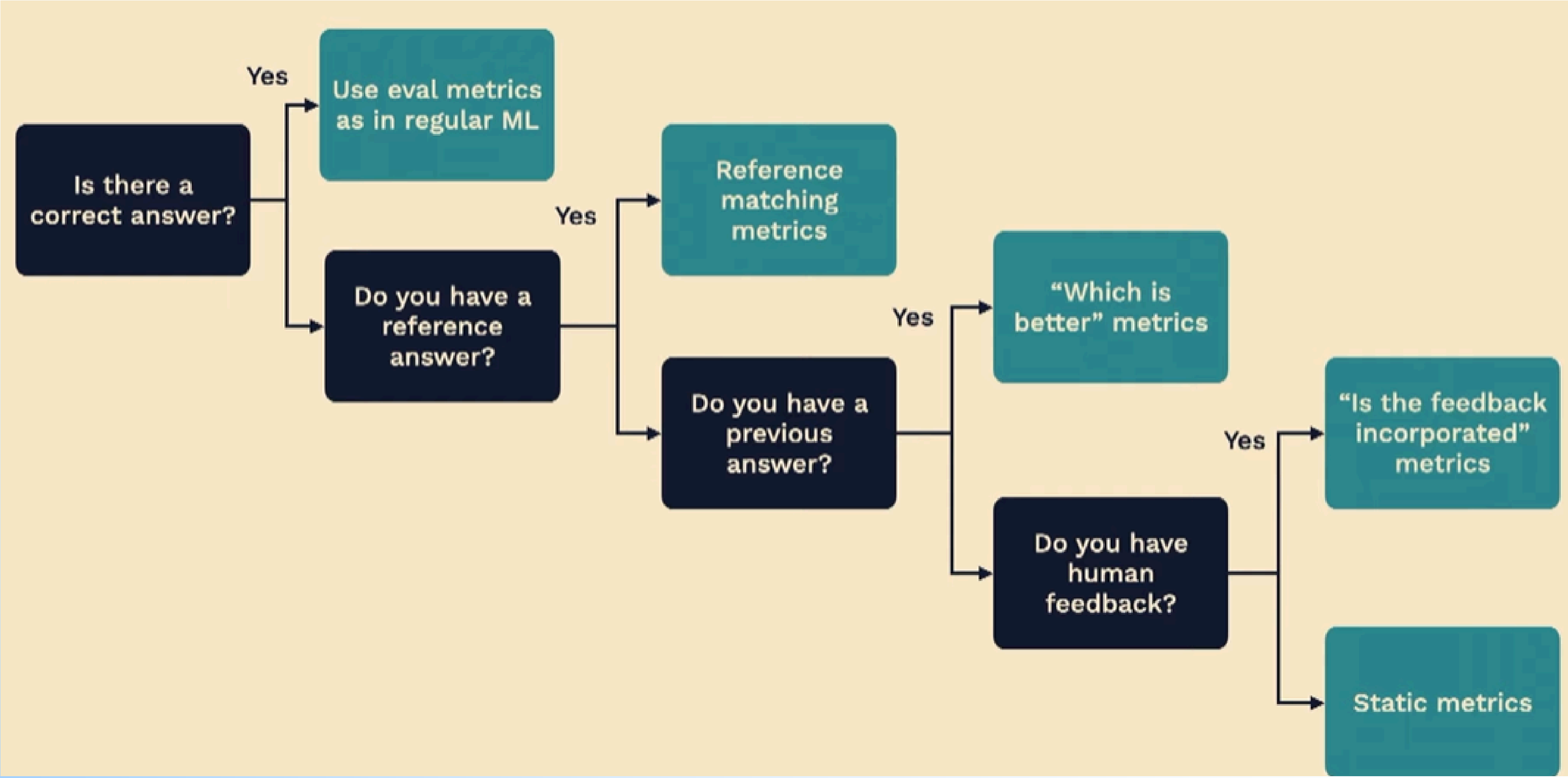
The problem with benchmarks

- Benchmark doesn’t work on your case
- Doesn’t include prompting, ICL, finetune etc.
- Measure issue above

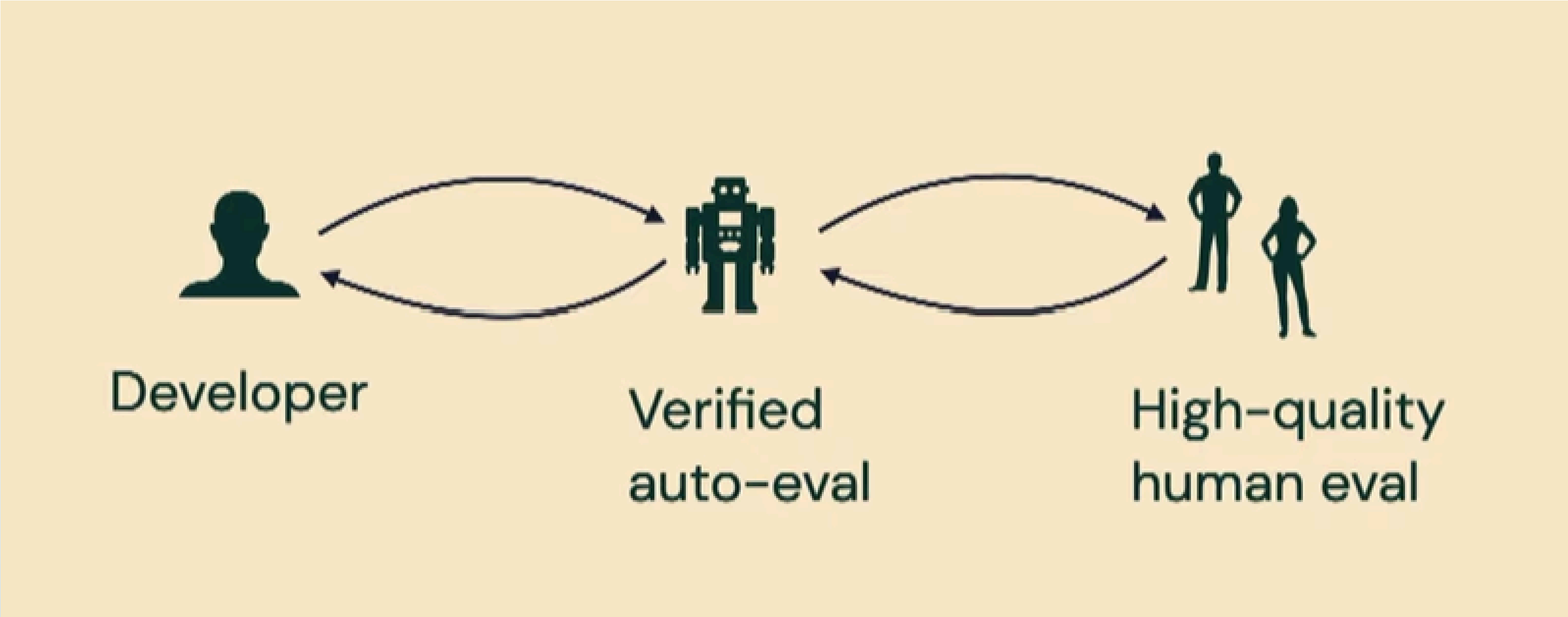
Building your own evaluation set

- Start incrementally
- Use your LLM to help
- Add more data as you roll out

Choosing evaluation metrics



The role Of human evaluation



Test Driver Workflow for LLM