



2022 AACL-IJCNLP

Recent Advances in Pre-trained Language Models: Why Do They Work and How to Use Them

姜成翰

Cheng-Han Chiang

National Taiwan University



莊永松

Yung-Sung Chuang

CSAIL, MIT



李宏毅

Hung-yi Lee

National Taiwan University





2022 AACL-IJCNLP

Recent Advances in **Pre-trained Language Models:** **Why Do They Work** and How to Use Them

17:00 – 17:10 **Part 1** Introduction

17:10 – 17:40 **Part 2** Why do PLMs work

Hung-yi Lee

National Taiwan University



Link to slides



2022 AACL-IJCNLP

Recent Advances in Pre-trained Language Models: Why Do They Work and **How To Use Them**

17:40 – 18:20

Part 3 How to Use PLMs: Contrastive learning
for Pre-trained Language Models

Yung-Sung Chuang
CSAIL, MIT



Link to slides





2022 AACL-IJCNLP

Recent Advances in Pre-trained Language Models: Why Do They Work and **How to Use Them**

18:40 – 19:50 **Part 4 + 5** How to Use PLMs: New fine-tuning
methods

Cheng-Han Chiang
National Taiwan University



Link to slides



Schedule

- 17:00 – 17:10 **Part 1** Introduction [Hung-yi]
- 17:10 – 17:40 **Part 2** Why do PLMs work [Hung-yi]
- 17:40 – 18:20 **Part 3** How to use PLMs: Contrastive Learning for PLMs [Yung-Sung]
- 18:20 – 18:30 Q&A for Part 1+2+3
- 18:30 – 18:40 Break
- 18:40 – 19:05 **Part 4** How to use PLMs: Parameter-efficient fine-tuning [Cheng-Han]
- 19:05 – 19:50 **Part 5** How to use PLMs: Using PLMs with different amounts of data [Cheng-Han]
- 19:50 – 20:00 Conclusion and Future work + Q&A



2022 AACL-IJCNLP

Part 1: Introduction

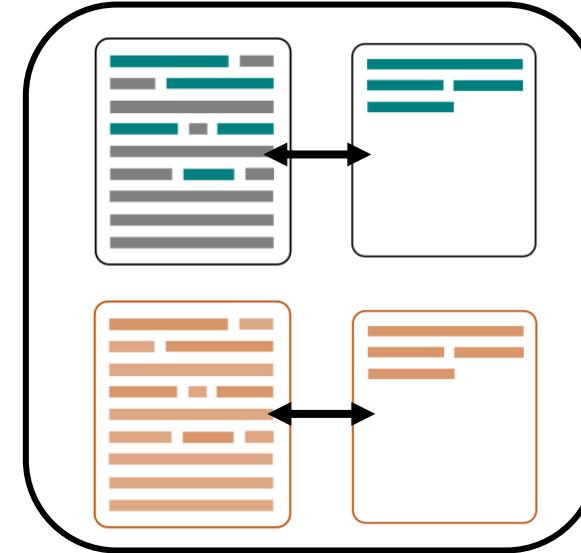
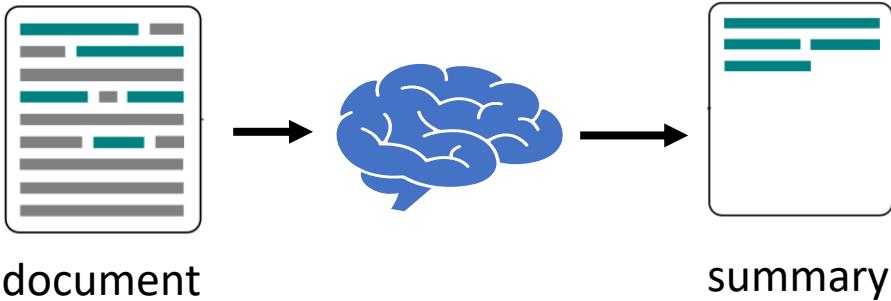
Hung-yi Lee

National Taiwan University

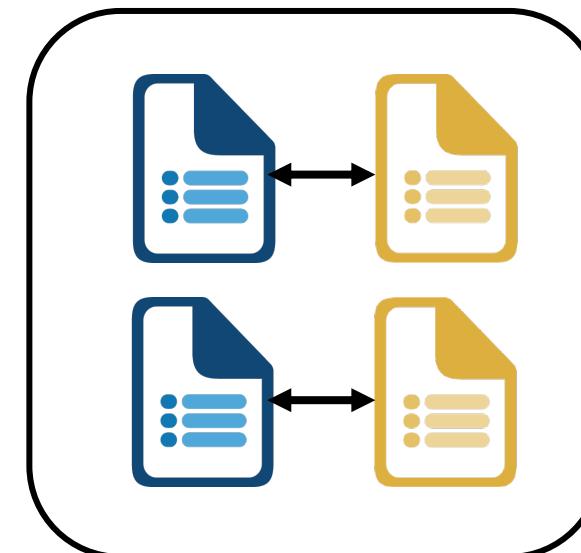
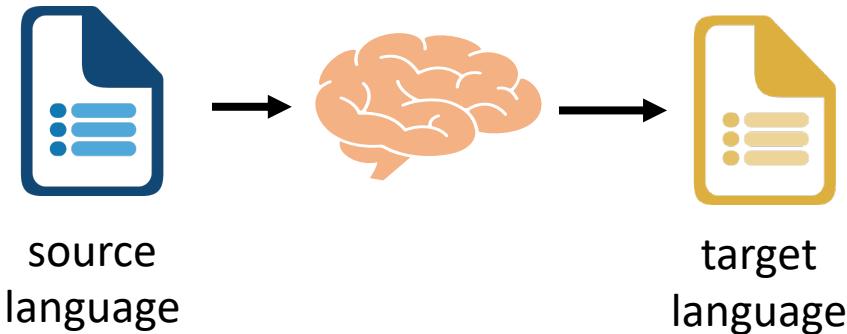


Deep Learning for Human Language Processing

Summarization



Machine Translation



So many different tasks

<https://youtu.be/tFBrqPPxWzE>

Speech Recognition

Speaker Recognition

Text-to-Speech (TTS)

Denoising

Speech Separation

Voice Conversion (VC)

Spoken Term Detection (STD)

Speech Question Answering

Speech Translation

Spoken Language Understanding

.....

Coreference Resolution

Syntactic Parsing

Semantic Parsing

Chatbot

Summarization

Text Style Transfer

Retrieval

Question Answering

Text Translation

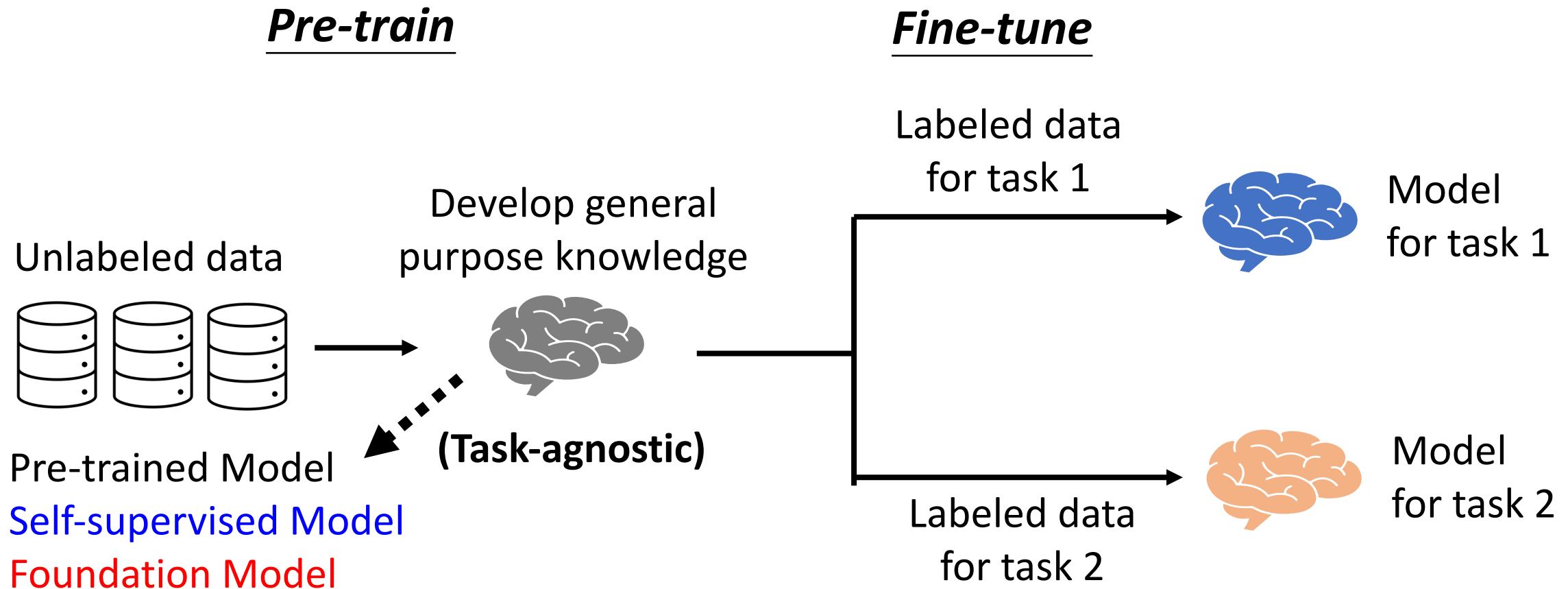
Dialogue State Tracking

.....

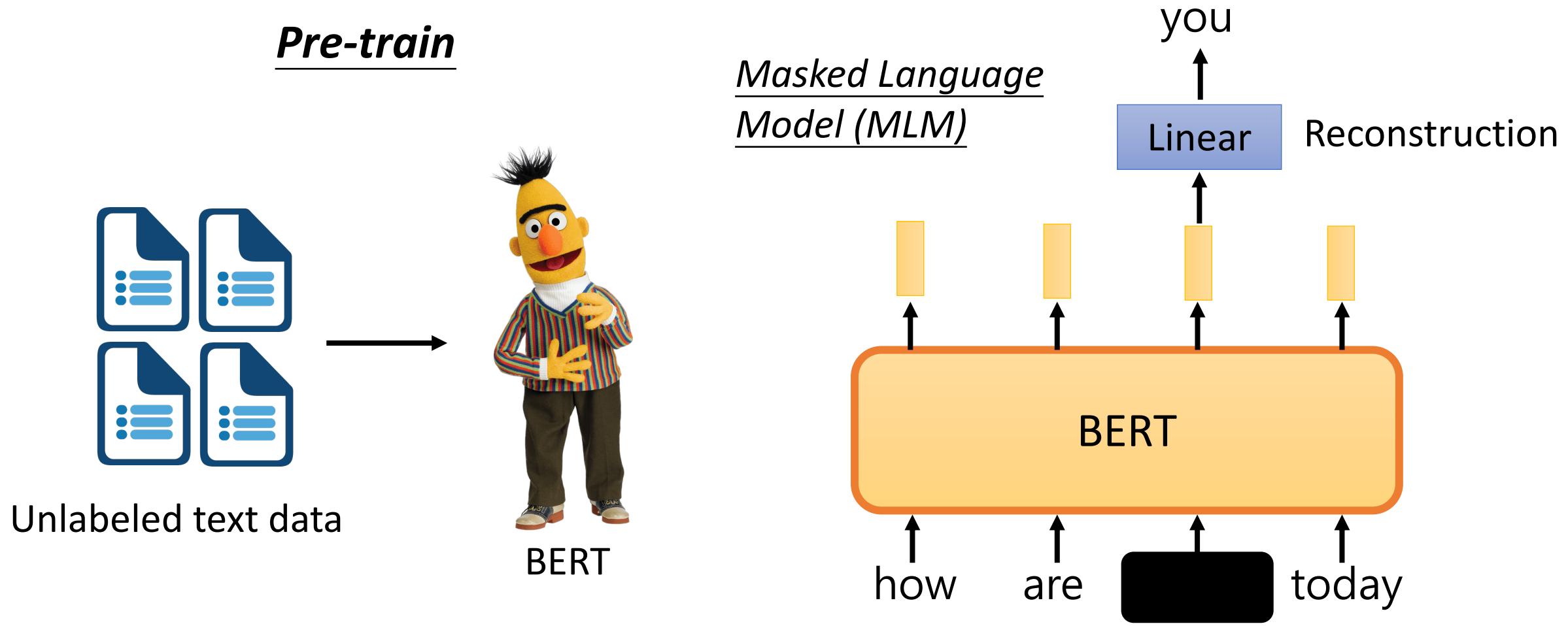
So many languages



Framework of Pre-training



Pre-training for NLP



This is not a complete survey!

Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, Graham Neubig

<https://arxiv.org/abs/2107.13586>

A Primer in BERTology: What we know about how BERT works

Anna Rogers, Olga Kovaleva, Anna Rumshisky

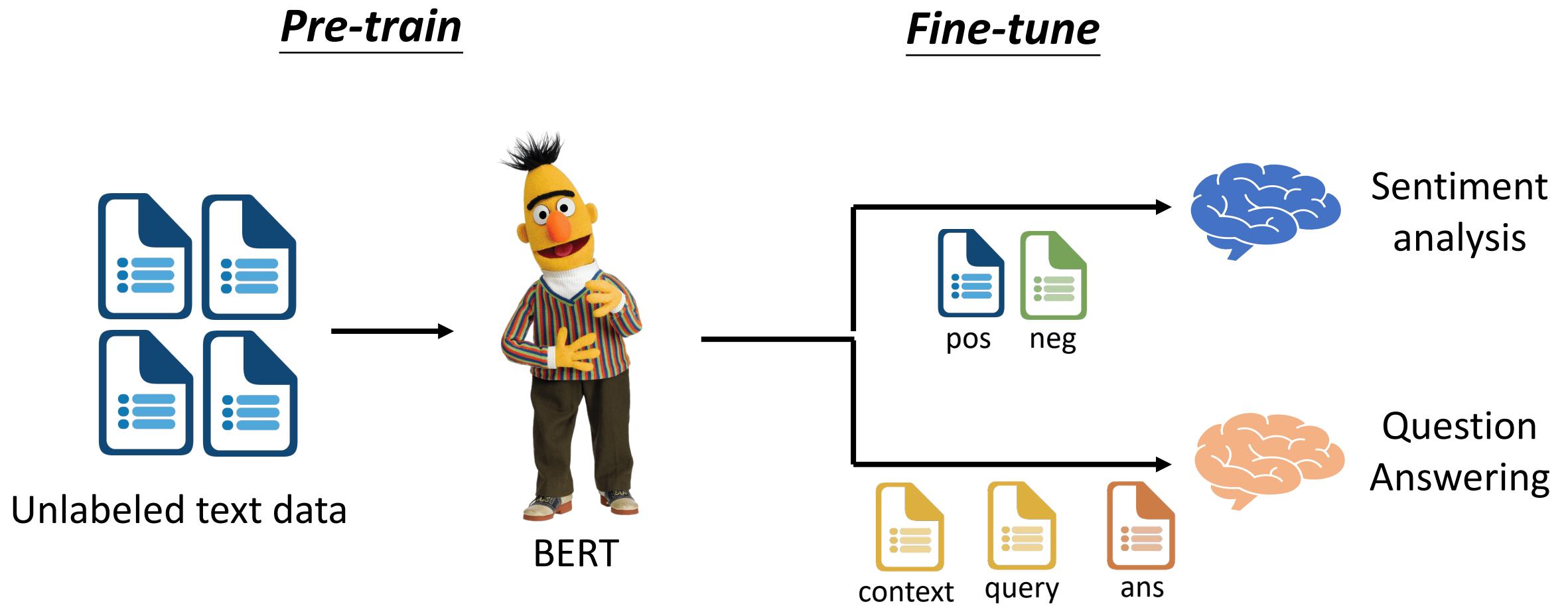
<https://arxiv.org/abs/2002.12327>

Pre-trained Models for Natural Language Processing: A Survey

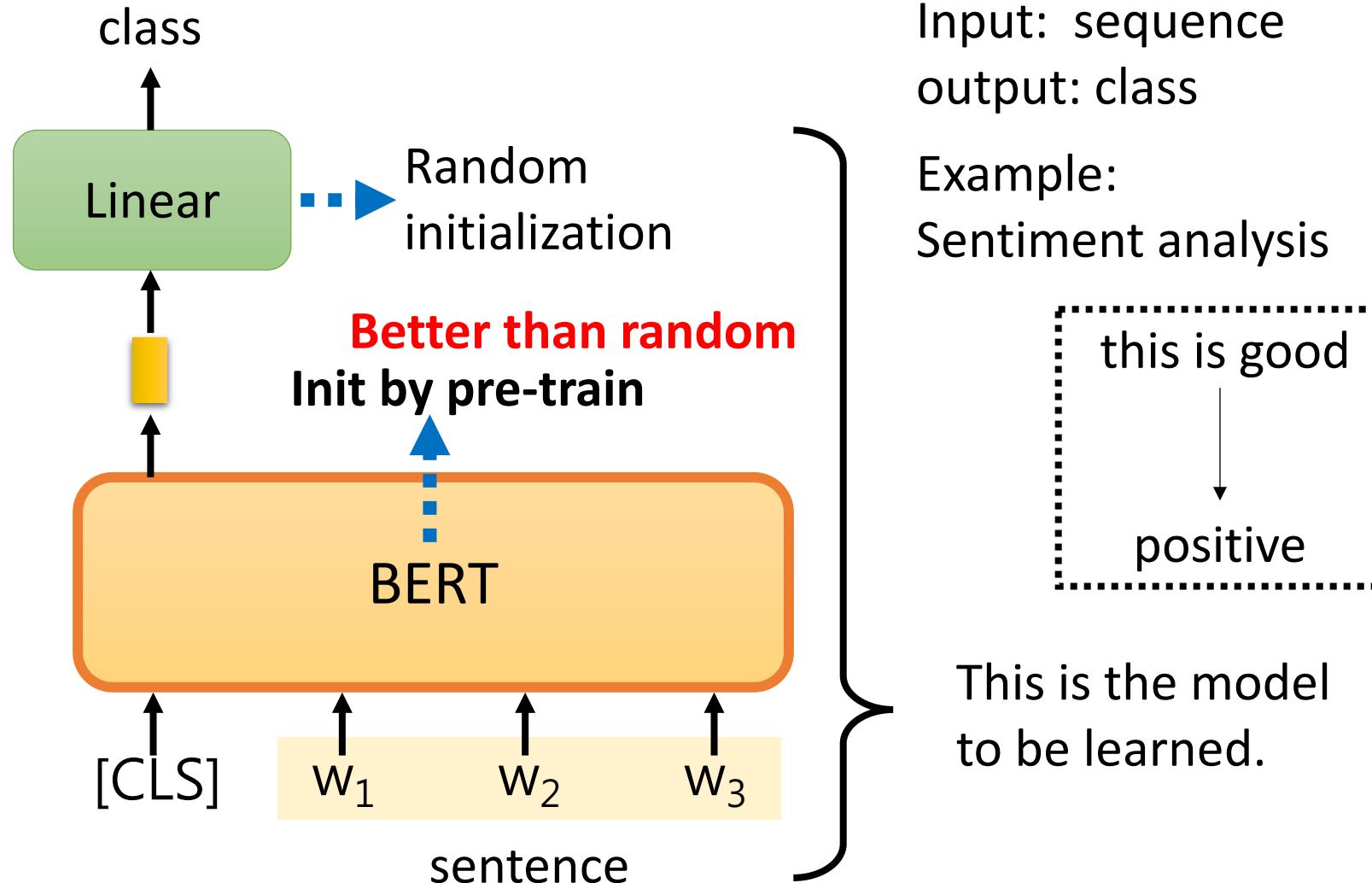
Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, Xuanjing Huang

<https://arxiv.org/abs/2003.08271>

Pre-training for NLP



Pre-training for NLP - Fine-tune (Example)



Pre-training for NLP

General Language Understanding Evaluation (GLUE)

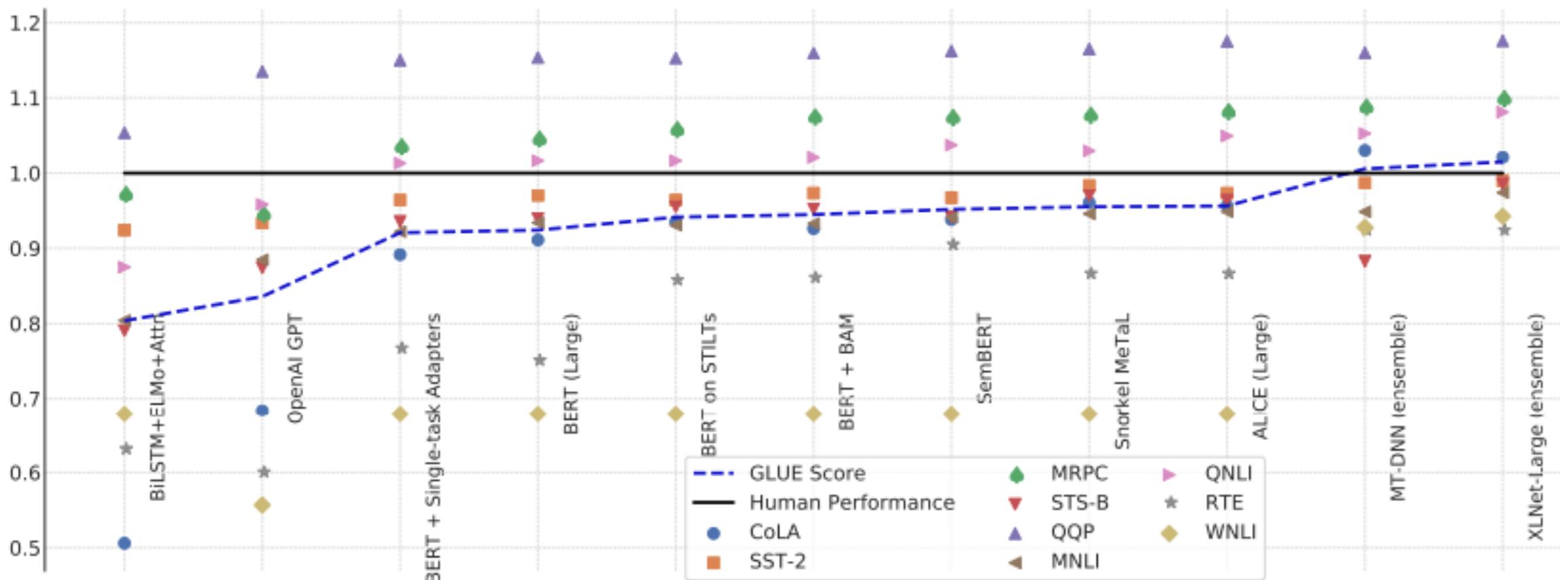
<https://gluebenchmark.com/>

- Corpus of Linguistic Acceptability (CoLA)
- Stanford Sentiment Treebank (SST-2)
- Microsoft Research Paraphrase Corpus (MRPC)
- Quora Question Pairs (QQP)
- Semantic Textual Similarity Benchmark (STS-B)
- Multi-Genre Natural Language Inference (MNLI)
- Question-answering NLI (QNLI)
- Recognizing Textual Entailment (RTE)
- Winograd NLI (WNLI)

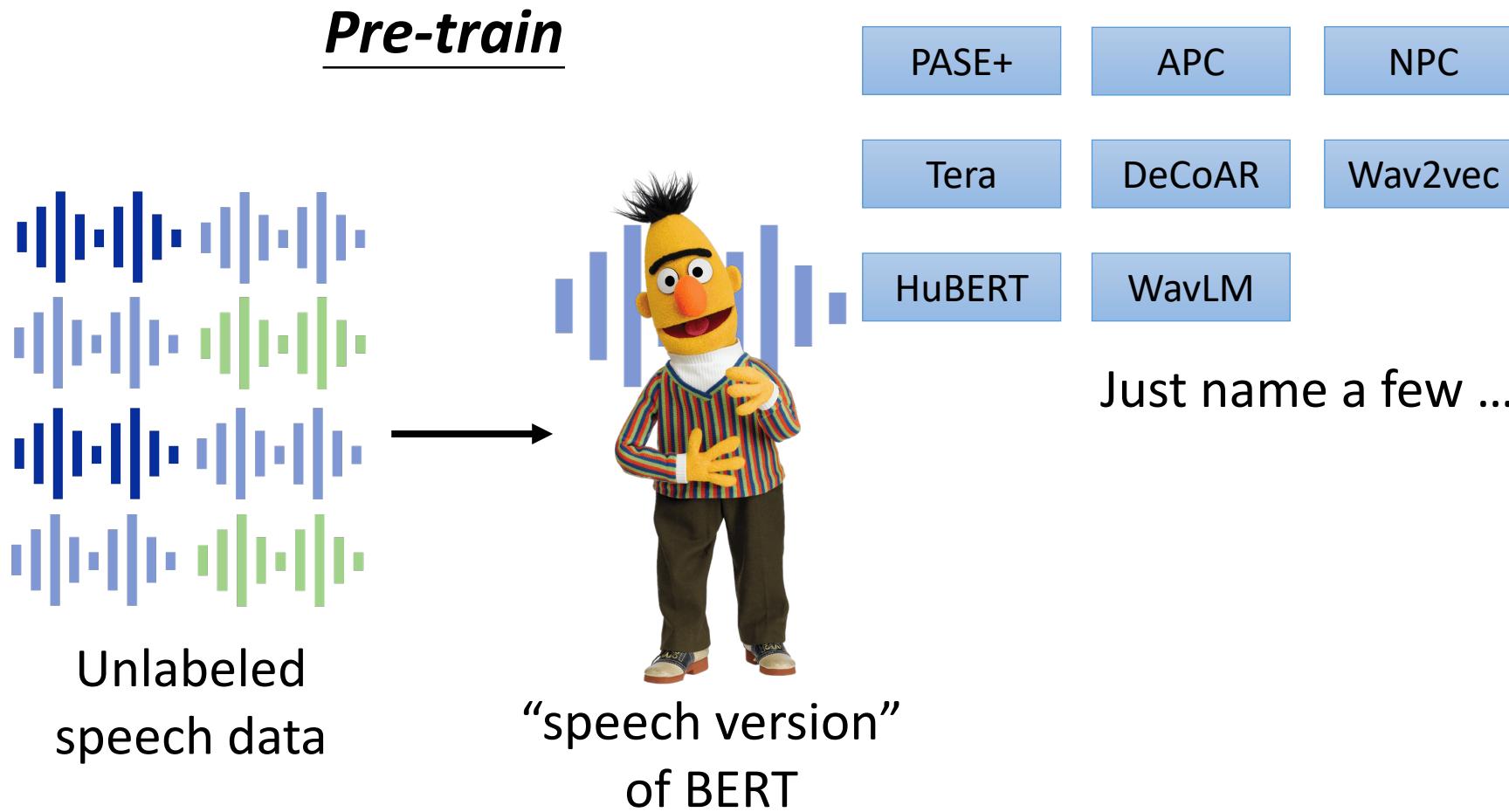


Pre-training for NLP

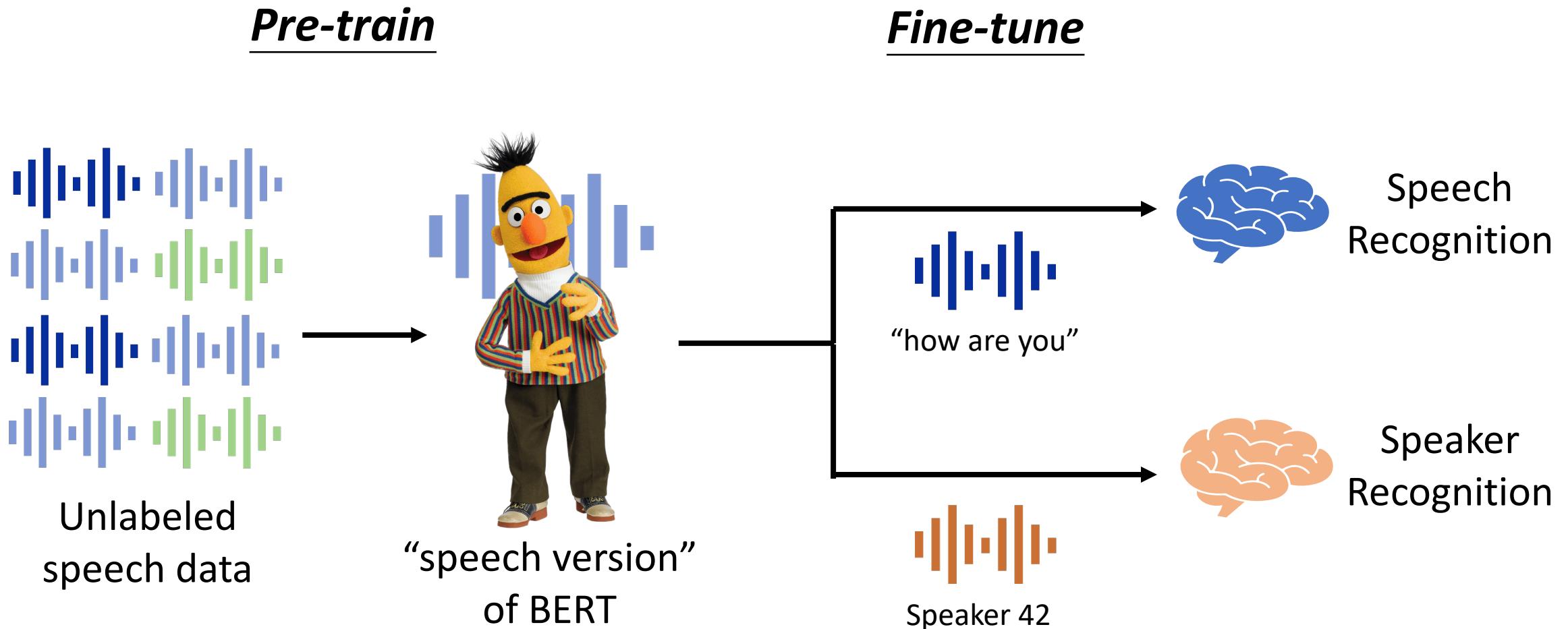
- GLUE scores



Pre-training for Speech



Pre-training for Speech



Speech processing Universal PERformance Benchmark (SUPERB)

<https://superbbenchmark.org/>

Phoneme
Recognition

Speaker
Identificaiton

Intent
Classifcaiton

Voice
Conversion



Keyword
Spotting

Speaker
Verificaiton

Spoken
Slot Filling

Speech
Enhancement



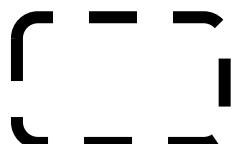
Published
at IS 2021

ASR

Speaker
Diarization

Speech
Translation

Speaker
Separation



QbyE

Emotion
Recognition

Speech QA

Published
at ACL 2022



Content



Speaker



Paralinguistic



Semantic



Synthesis



Published
at IS 2022

SUPERB: Speech processing Universal PERformance Benchmark

Shu-wen Yang¹, Po-Han Chi^{1}, Yung-Sung Chuang^{1*}, Cheng-I Jeff Lai^{2*}, Kushal Lakhotia^{3*},
Yist Y. Lin^{1*}, Andy T. Liu^{1*}, Jiatong Shi^{4*}, Xuankai Chang⁶, Guan-Ting Lin¹,
Tzu-Hsien Huang¹, Wei-Cheng Tseng¹, Ko-tik Lee¹, Da-Rong Liu¹, Zili Huang⁴, Shuyan Dong^{5†},
Shang-Wen Li^{5†}, Shinji Watanabe⁶, Abdelrahman Mohamed³, Hung-yi Lee¹*

Presented at INTERSPEECH 2021

<https://arxiv.org/abs/2105.01051>



SUPERB-SG: Enhanced Speech processing Universal PERformance Benchmark for Semantic and Generative Capabilities

**Hsiang-Sheng Tsai^{1*}, Heng-Jui Chang^{1*}, Wen-Chin Huang^{2*}, Zili Huang^{3*}, Kushal Lakhotia^{4*},
Shu-wen Yang¹, Shuyan Dong⁵, Andy T. Liu¹, Cheng-I Lai⁶,
Jiatong Shi⁷, Xuankai Chang⁷, Phil Hall⁸, Hsuan-Jui Chen¹,
Shang-Wen Li⁵, Shinji Watanabe⁷, Abdelrahman Mohamed⁵, Hung-yi Lee¹**

Presented at ACL 2022

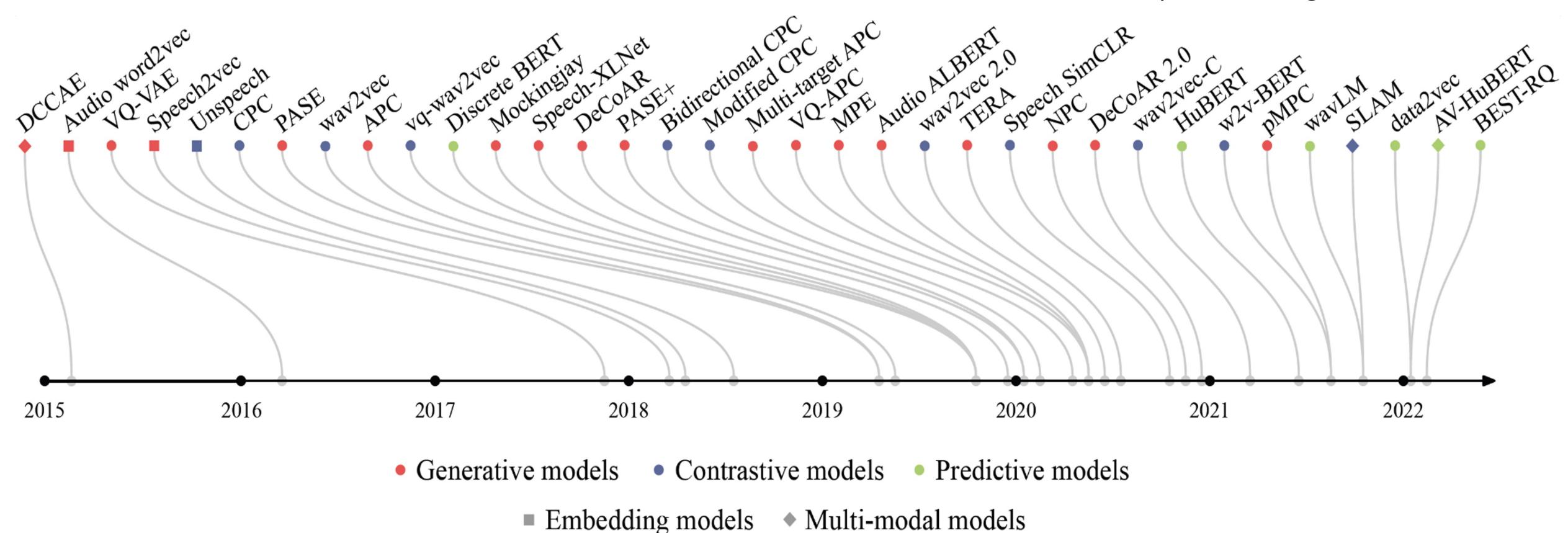
<https://arxiv.org/abs/2203.06849>

<https://youtu.be/GTjwYzFG54E>

Self-Supervised Speech Representation Learning: A Review

Abdelrahman Mohamed*, Hung-yi Lee*, Lasse Borgholt*, Jakob D. Havnør*, Joakim Edin, Christian Igel
Katrin Kirchhoff, Shang-Wen Li, Karen Livescu, Lars Maaløe, Tara N. Sainath, Shinji Watanabe

<https://arxiv.org/abs/2205.10643>



Schedule

- 17:00 – 17:10 **Part 1** Introduction [Hung-yi]
- 17:10 – 17:40 **Part 2** Why do PLMs work [Hung-yi]
- 17:40 – 18:20 **Part 3** How to use PLMs: Contrastive Learning for PLMs [Yung-Sung]
- 18:20 – 18:30 Q&A for Part 1+2+3
- 18:30 – 18:40 Break
- 18:40 – 19:05 **Part 4** How to use PLMs: Parameter-efficient fine-tuning [Cheng-Han]
- 19:05 – 19:50 **Part 5** How to use PLMs: Using PLMs with different amounts of data [Cheng-Han]
- 19:50 – 20:00 Conclusion and Future work + Q&A



2022 AACL-IJCNLP

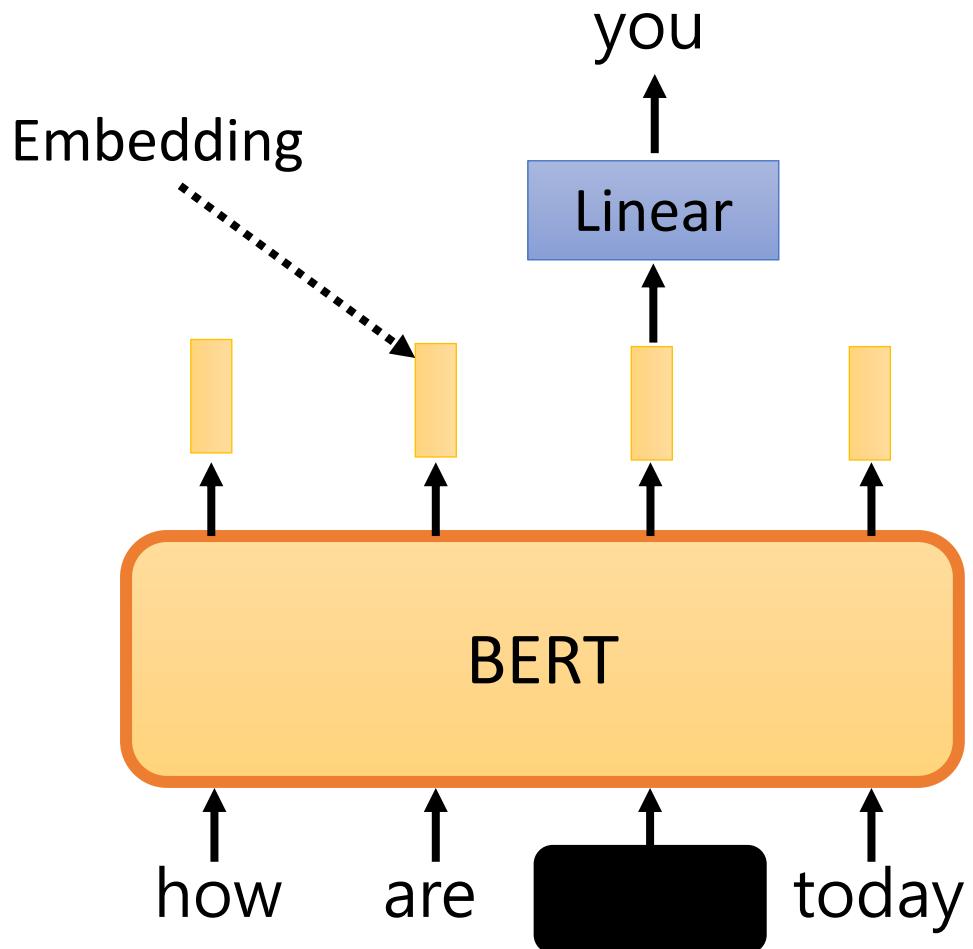
Part 2: Why do PLMs work

Hung-yi Lee

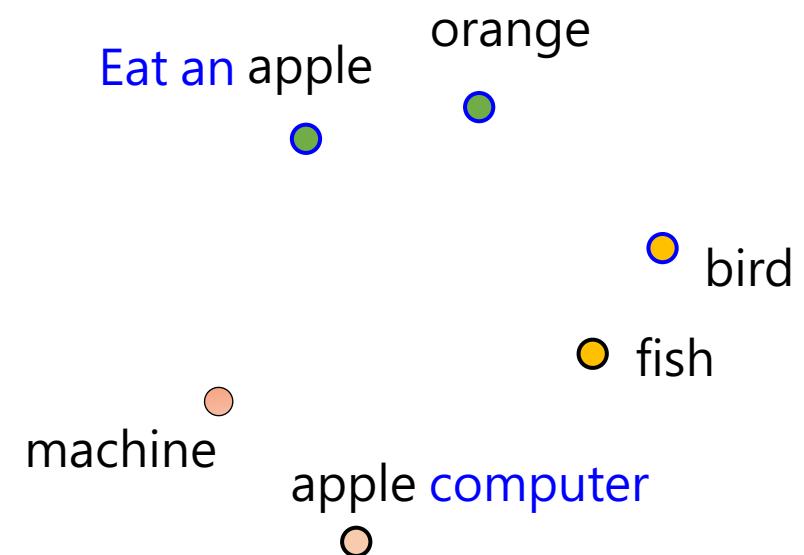
National Taiwan University



Contextualized Word Representations



The tokens with similar meaning have similar embedding.



Context is considered.

“Lie” clusters

Mathematical sense - verb

- A skew polygon does not **lie** in a flat plane, but zigzags in three (or more) dimensions
- As an open string propagates through spacetime, its endpoints are required to **lie** on a D-brane..

Untruth - verb

- Take for example the declaration "I will **lie** for personal benefit."
- Rob reveals to Tracy that everything was a **lie** and that he still hated her.

Lie down - verb

- There Fenrir will **lie** until Ragnarok.
- They **lie** down to sleep deeply

Geographical (island) - verb

Some 3,579 islands **lie** adjacent to the peninsula.
The islands **lie** on the Kerguelen Plateau in the Indian Ocean.

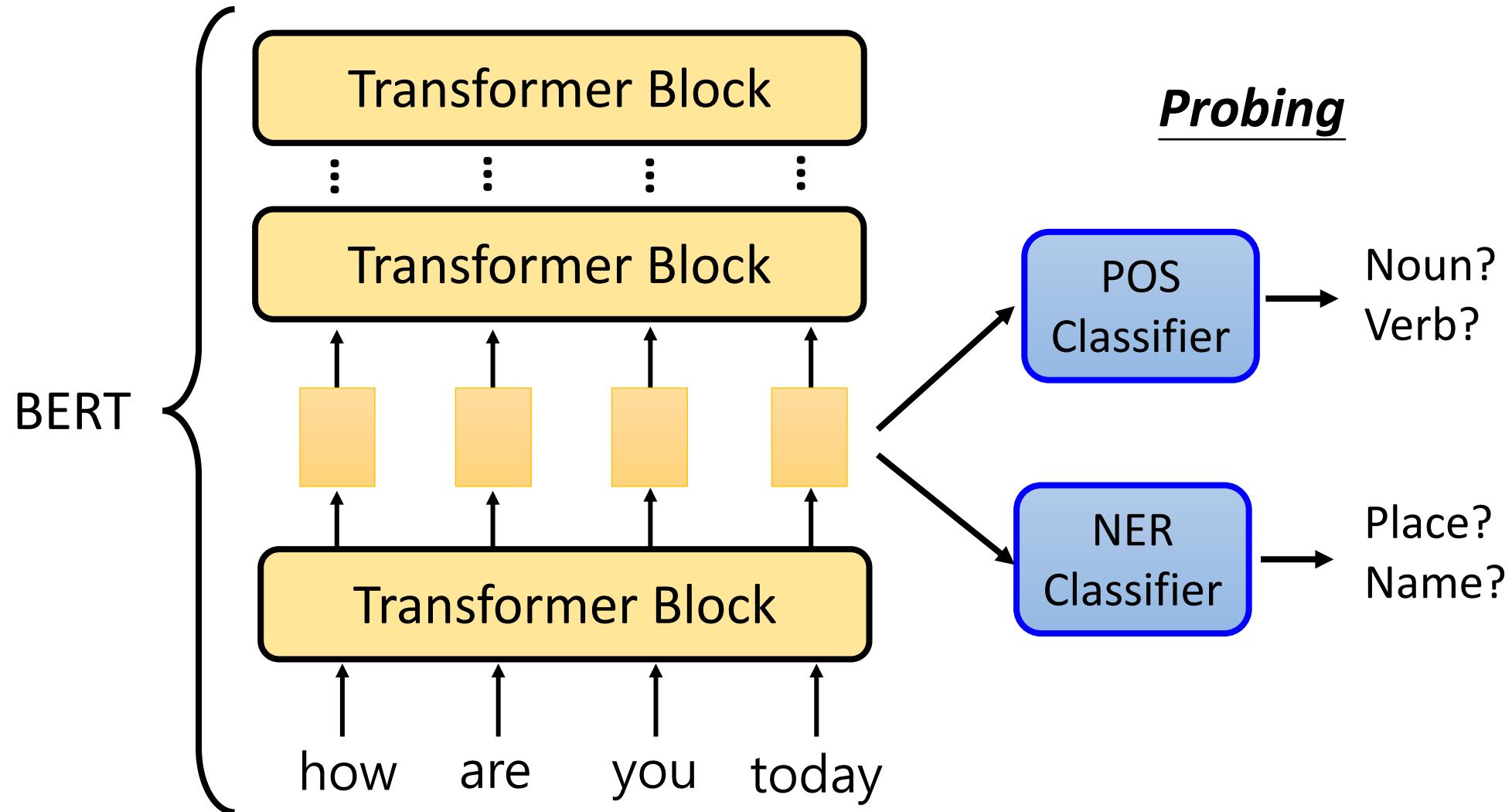
Conceptual placement - verb

- According to Dewey, conversation, debate and dialogue **lie** at the heart of a democracy
- The origins of mathematical thought **lie** in the concepts of number, magnitude and form.

Geographical (other) - verb

Very small portions **lie** within the Pueblo County School District 70.
The ruins of the town **lie** along the river Ziz in the Tafilalt oasis near the town of Rissani.

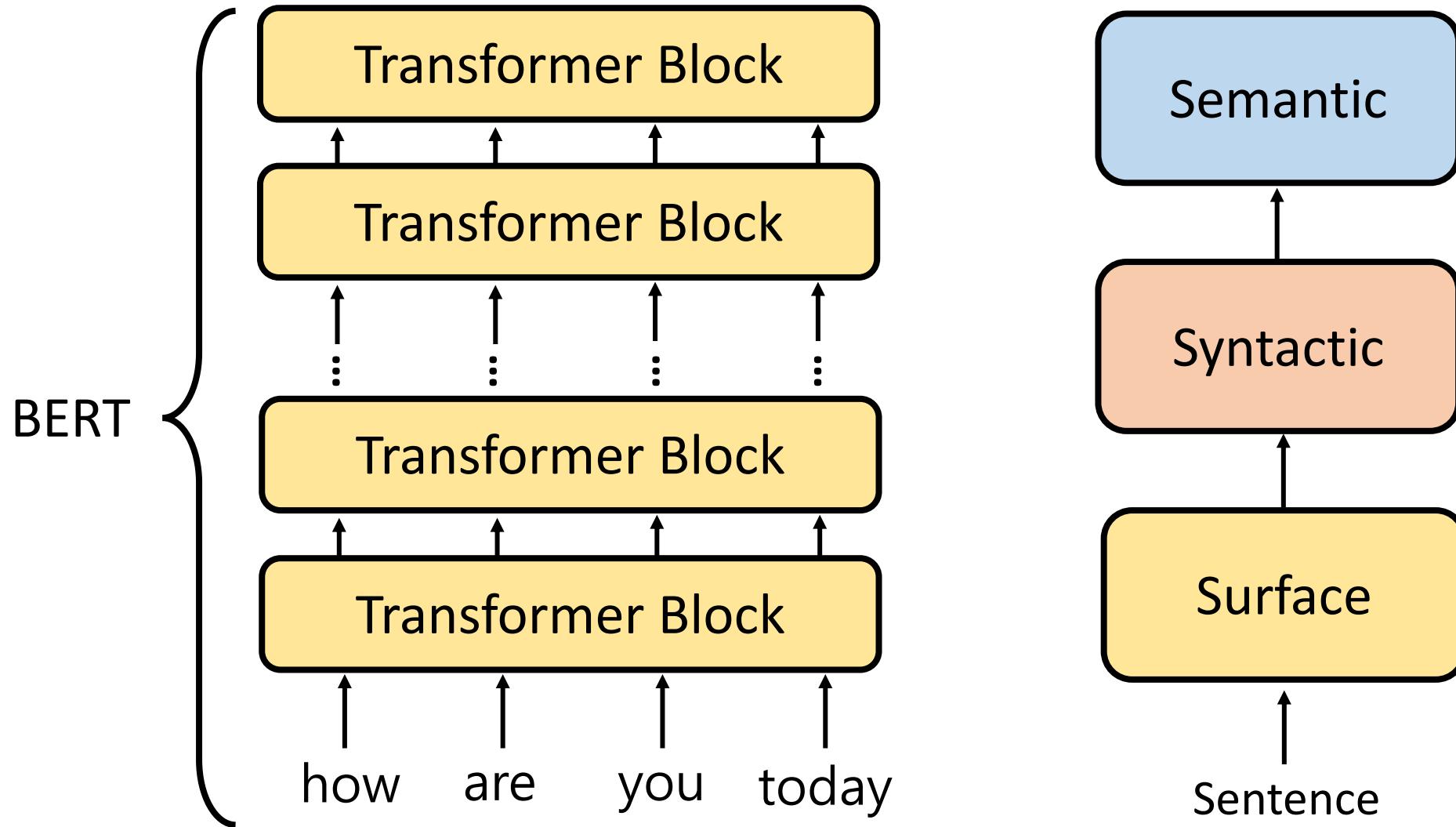
BERTology - What does each layer learn?



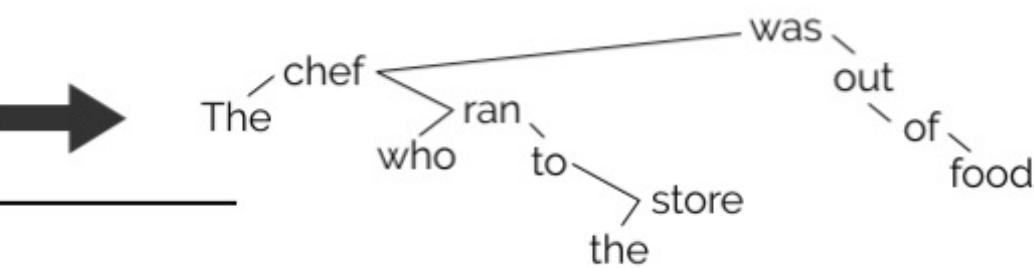
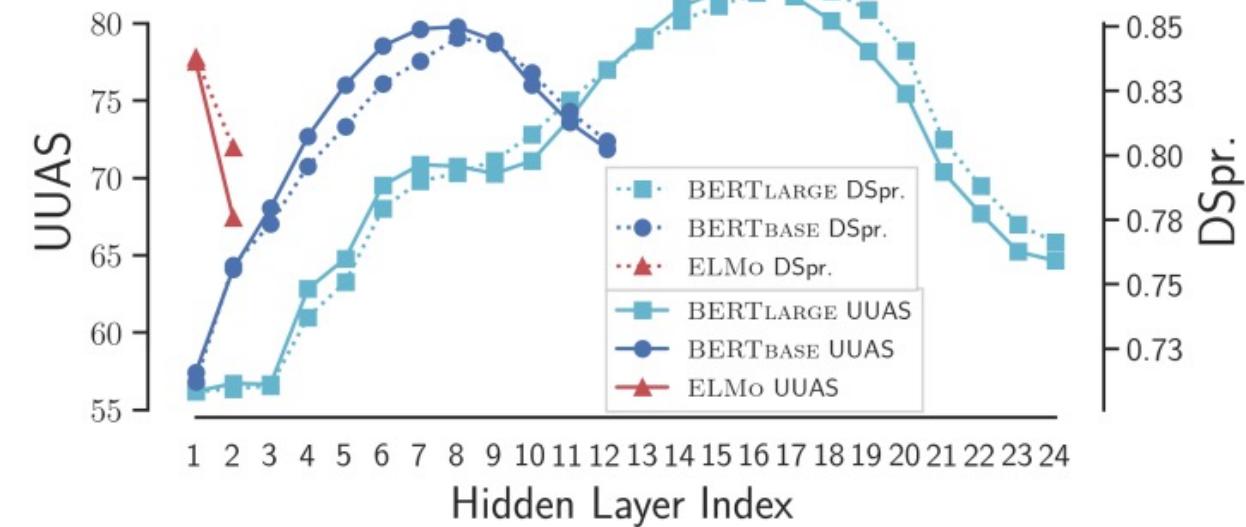
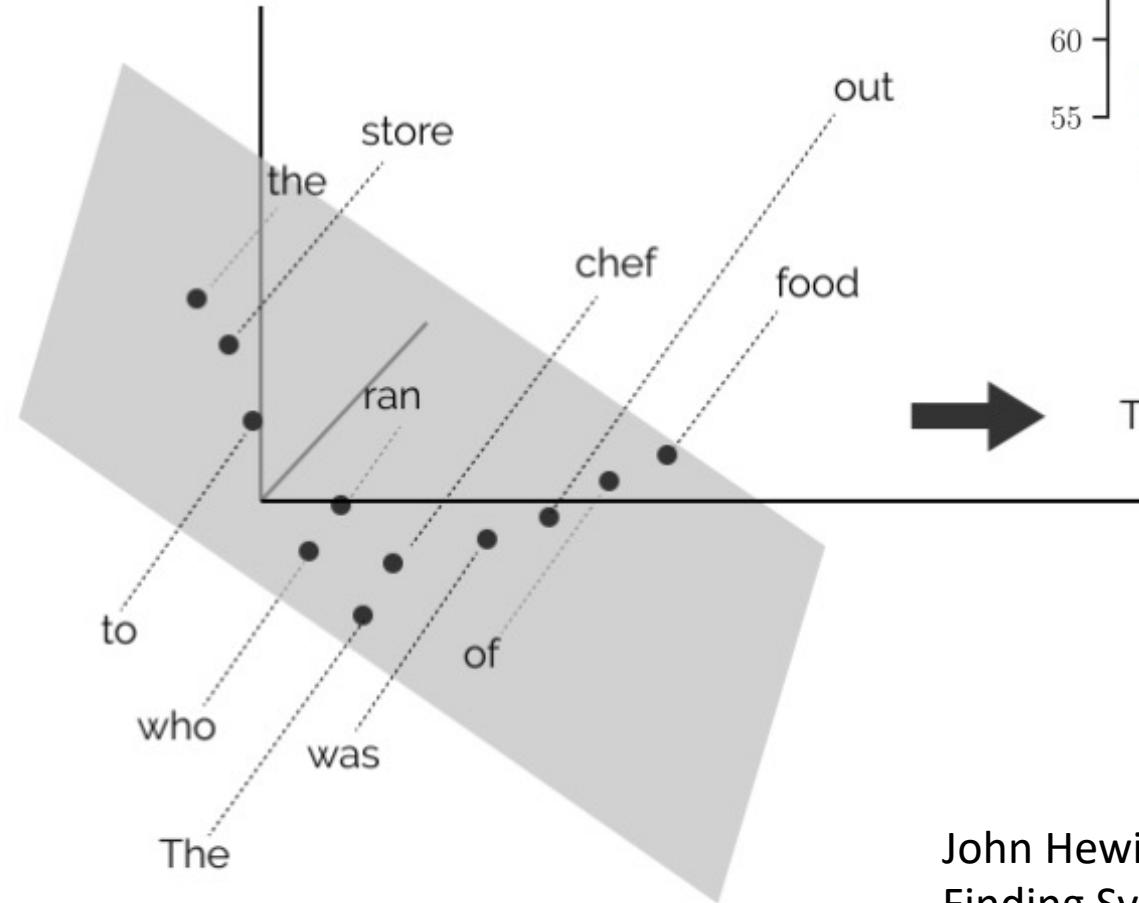
BERTology - What does each layer learn?

- Higher classifier accuracy does not always mean encoding more information.
- Interpret the prob results with care 😊
 - John Hewitt, Percy Liang, Designing and Interpreting Probes with Control Tasks, EMNLP, 2019
 - Elena Voita, Ivan Titov, Information-Theoretic Probing with Minimum Description Length, EMNLP, 2020
 - John Hewitt, Kawin Ethayarajh, Percy Liang, Christopher Manning, Conditional probing: measuring usable information beyond a baseline, ENMLP, 2021
 - Jiaoda Li, Ryan Cotterell, Mrinmaya Sachan, Probing via Prompting, NAACL, 2022

BERTology - What does each layer learn?



Encoding Syntax

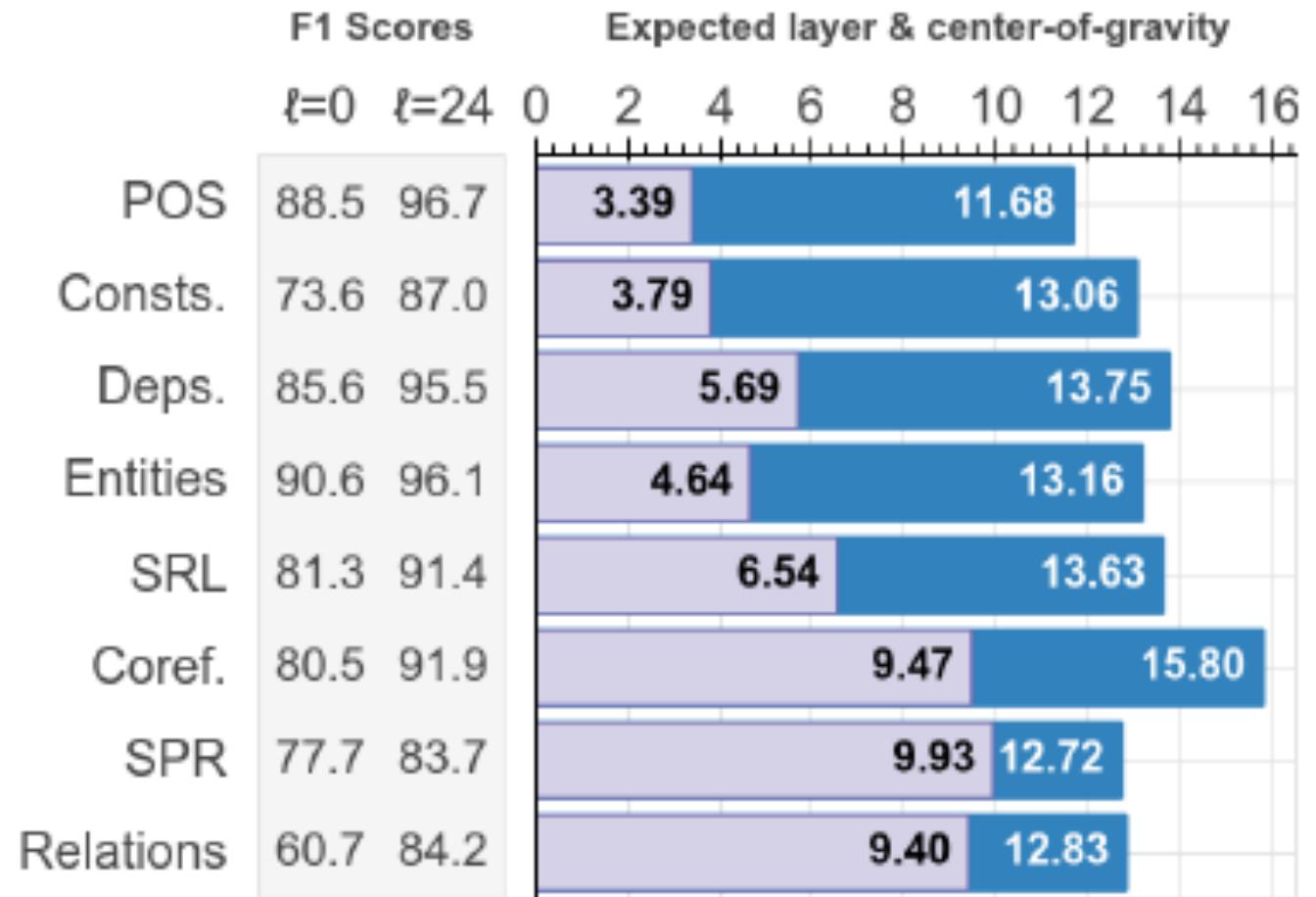


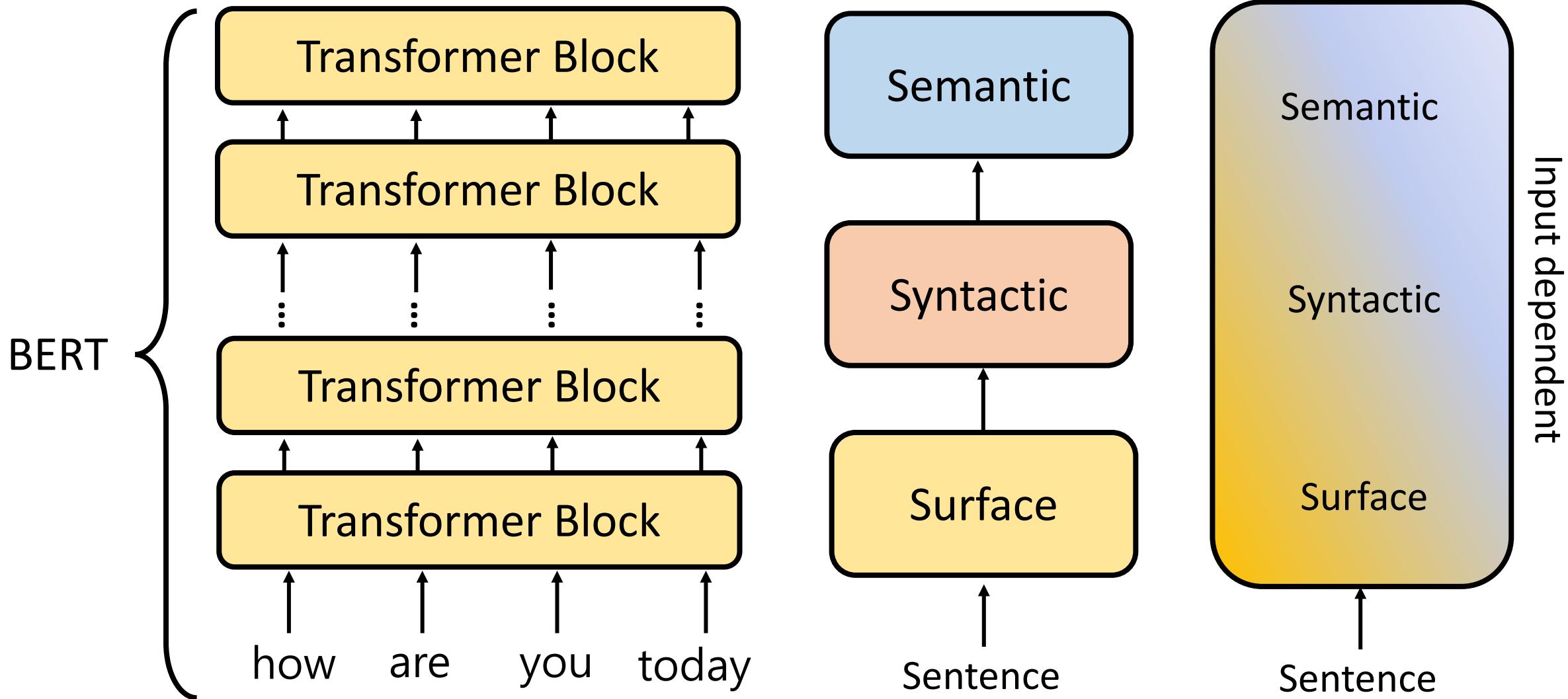
John Hewitt, Christopher D. Manning, A Structural Probe for Finding Syntax in Word Representations, NAACL, 2019

BERTology - What does each layer learn?

Layer	SentLen (Surface)	WC (Surface)	TreeDepth (Syntactic)	TopConst (Syntactic)	BShift (Syntactic)	Tense (Semantic)	SubjNum (Semantic)	ObjNum (Semantic)	SOMO (Semantic)	CoordInv (Semantic)
1	93.9 (2.0)	24.9 (24.8)	35.9 (6.1)	63.6 (9.0)	50.3 (0.3)	82.2 (18.4)	77.6 (10.2)	76.7 (26.3)	49.9 (-0.1)	53.9 (3.9)
2	95.9 (3.4)	65.0 (64.8)	40.6 (11.3)	71.3 (16.1)	55.8 (5.8)	85.9 (23.5)	82.5 (15.3)	80.6 (17.1)	53.8 (4.4)	58.5 (8.5)
3	96.2 (3.9)	66.5 (66.0)	39.7 (10.4)	71.5 (18.5)	64.9 (14.9)	86.6 (23.8)	82.0 (14.6)	80.3 (16.6)	55.8 (5.9)	59.3 (9.3)
4	94.2 (2.3)	69.8 (69.6)	39.4 (10.8)	71.3 (18.3)	74.4 (24.5)	87.6 (25.2)	81.9 (15.0)	81.4 (19.1)	59.0 (8.5)	58.1 (8.1)
5	92.0 (0.5)	69.2 (69.0)	40.6 (11.8)	81.3 (30.8)	81.4 (31.4)	89.5 (26.7)	85.8 (19.4)	81.2 (18.6)	60.2 (10.3)	64.1 (14.1)
6	88.4 (-3.0)	63.5 (63.4)	41.3 (13.0)	83.3 (36.6)	82.9 (32.9)	89.8 (27.6)	88.1 (21.9)	82.0 (20.1)	60.7 (10.2)	71.1 (21.2)
7	83.7 (-7.7)	56.9 (56.7)	40.1 (12.0)	84.1 (39.5)	83.0 (32.9)	89.9 (27.5)	87.4 (22.2)	82.2 (21.1)	61.6 (11.7)	74.8 (24.9)
8	82.9 (-8.1)	51.1 (51.0)	39.2 (10.3)	84.0 (39.5)	83.9 (33.9)	89.9 (27.6)	87.5 (22.2)	81.2 (19.7)	62.1 (12.2)	76.4 (26.4)
9	80.1 (-11.1)	47.9 (47.8)	38.5 (10.8)	83.1 (39.8)	87.0 (37.1)	90.0 (28.0)	87.6 (22.9)	81.8 (20.5)	63.4 (13.4)	78.7 (28.9)
10	77.0 (-14.0)	43.4 (43.2)	38.1 (9.9)	81.7 (39.8)	86.7 (36.7)	89.7 (27.6)	87.1 (22.6)	80.5 (19.9)	63.3 (12.7)	78.4 (28.1)
11	73.9 (-17.0)	42.8 (42.7)	36.3 (7.9)	80.3 (39.1)	86.8 (36.8)	89.9 (27.8)	85.7 (21.9)	78.9 (18.6)	64.4 (14.5)	77.6 (27.9)
12	69.5 (-21.4)	49.1 (49.0)	34.7 (6.9)	76.5 (37.2)	86.4 (36.4)	89.5 (27.7)	84.0 (20.2)	78.7 (18.4)	65.2 (15.3)	74.9 (25.4)

BERTology - What does each layer learn?





- Jingcheng Niu, Wenjie Lu, Gerald Penn, Does BERT Rediscover a Classical NLP Pipeline?, COLING, 2022
- Wietse de Vries, Andreas van Cranenburgh, Malvina Nissim, What's so special about BERT's layers? A closer look at the NLP pipeline in monolingual and multilingual models, EMNLP Finding, 2020

BERT Embryology

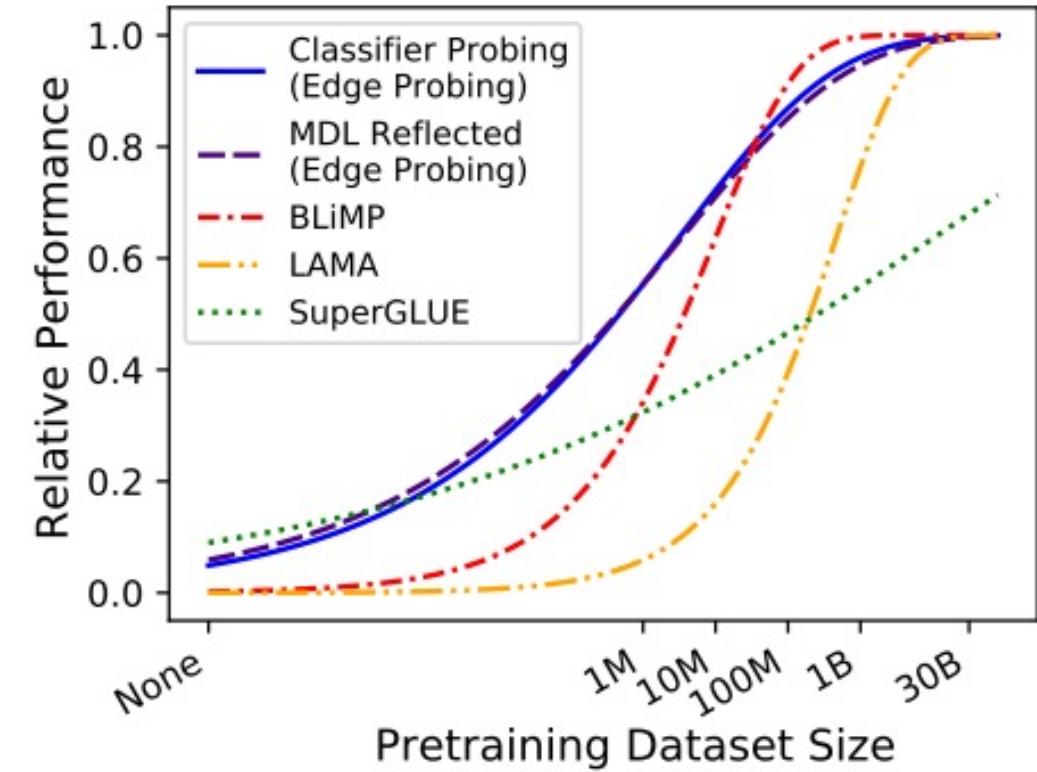
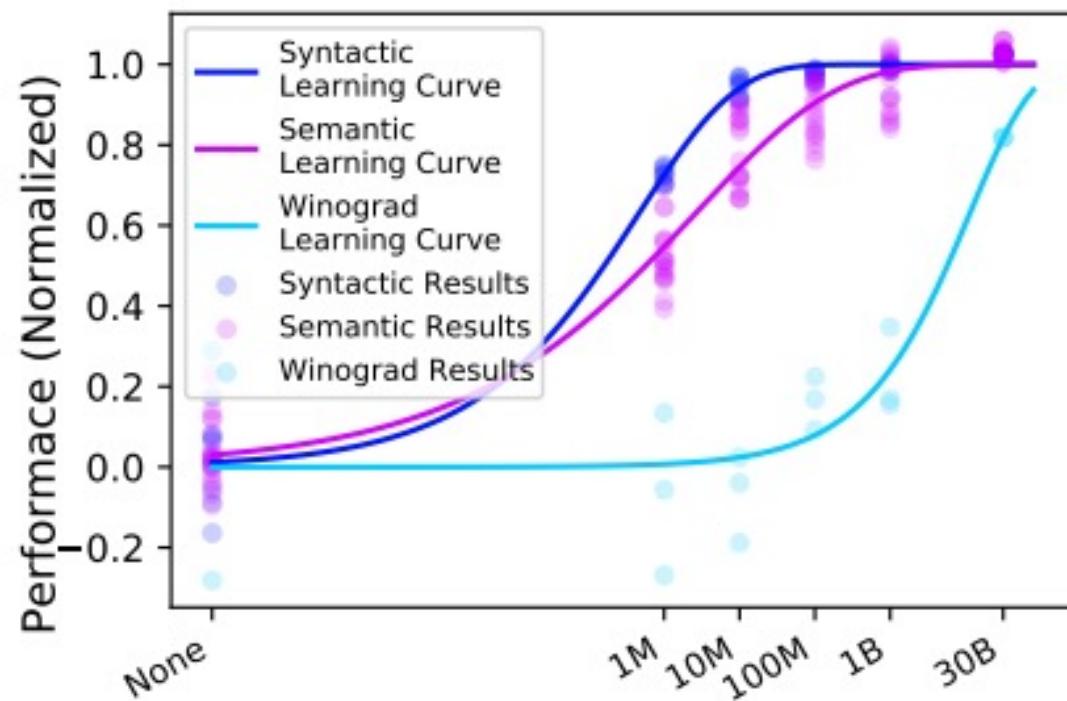
Analyzing what BERT learned during training

- Cheng-Han Chiang, Sung-Feng Huang, Hung-yi Lee, Pretrained Language Model Embryology: The Birth of ALBERT, EMNLP, 2020
- Leo Z. Liu, Yizhong Wang, Jungo Kasai, Hannaneh Hajishirzi, Noah A. Smith, Probing Across Time: What Does RoBERTa Know and When? EMNLP-finding, 2021



When does BERT know POS tagging, syntactic parsing, semantics?

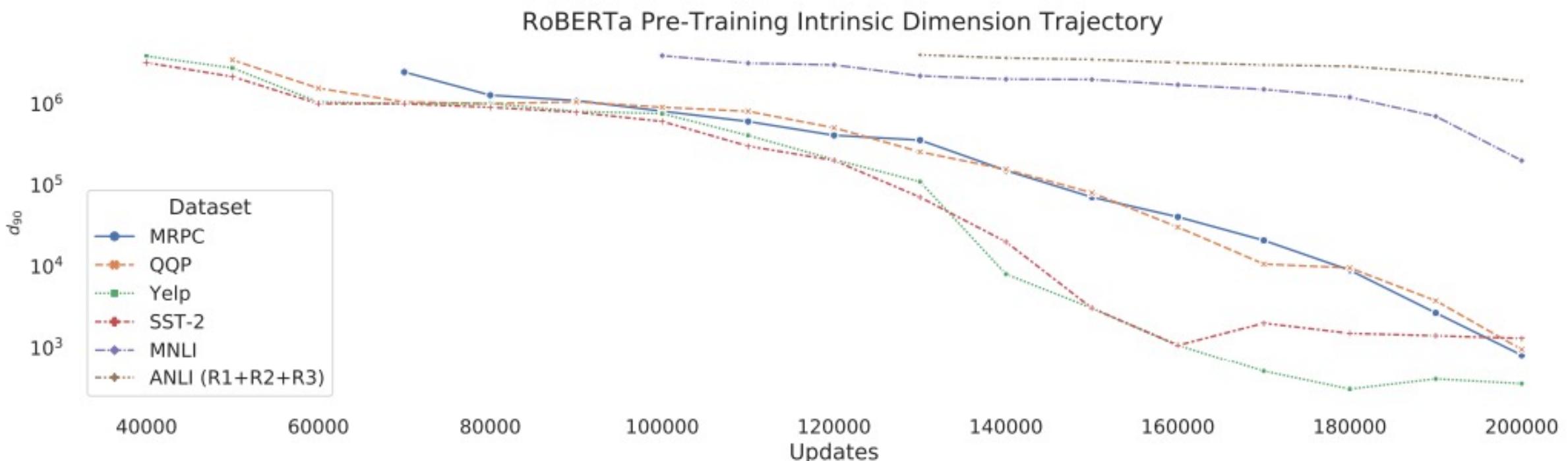
When Do You Need Billions of Words of Pretraining Data?



Yian Zhang, Alex Warstadt, Xiaocheng Li, Samuel R. Bowman, When Do You Need Billions of Words of Pretraining Data? ACL 2021

Pre-trained Intrinsic Dimension

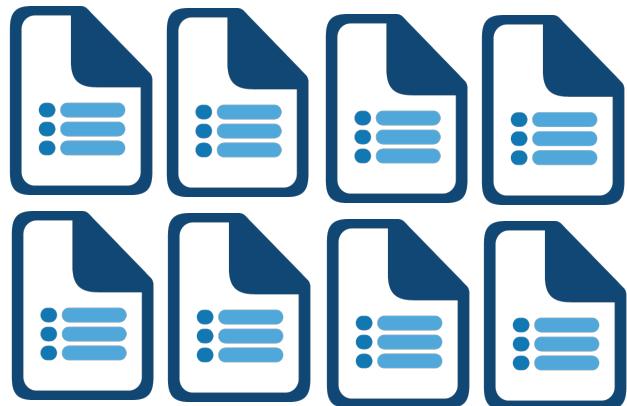
- Smaller intrinsic dimension means better generalization



Armen Aghajanyan, Sonal Gupta, Luke Zettlemoyer, Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning, ACL, 2021

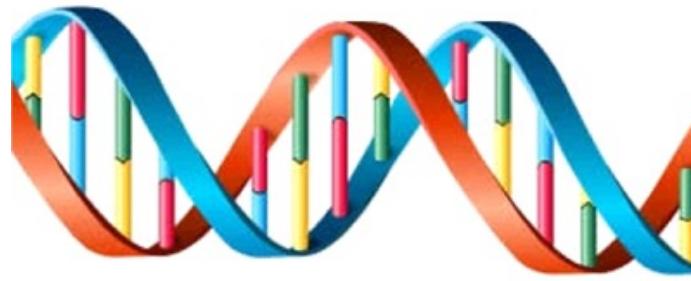
Cross-discipline Capability

Pre-train



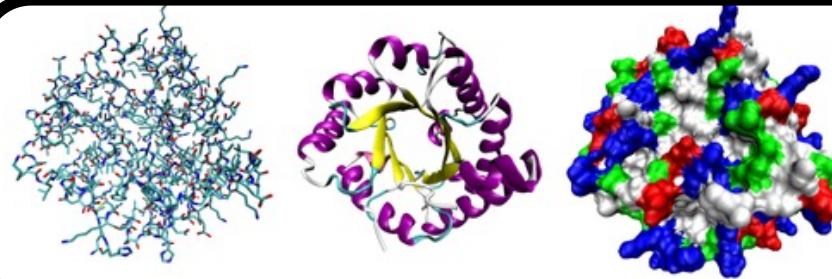
Human Language

Fine-tune



Testing

DNA
Classification



Protein
Classification

Wei-Tsung Kao, Hung-yi Lee, Is BERT a Cross-Disciplinary Knowledge Learner? A Surprising Finding of Pre-trained Models' Transferability, EMNLP finding, 2021

Cross-discipline Capability



Downstream task

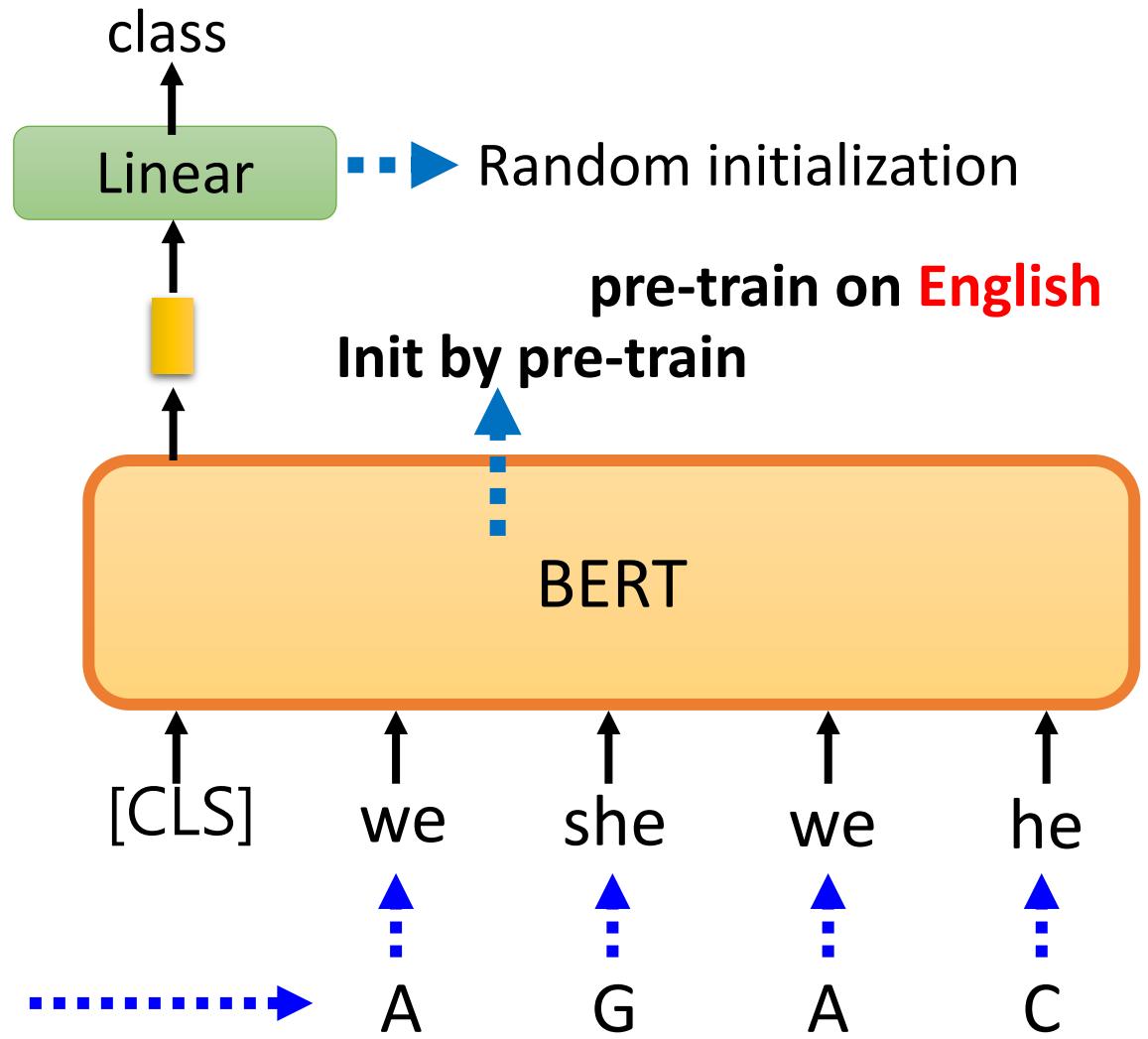
class	DNA sequence
EI	CCAGCTGCATCACAGGAGGCCAGCGAGCAGGTCTGTTCCAAGC
EI	AGACCCGCCGGGAGGCGGGAGGACCTGCAGGGTGAGCCCCACCC
IE	AACGTGGCCTCCTTGTGCCCTTCCCCACAGTGCCCTTCCAGG
IE	CCACTCAGCCAGGCCCTTCTTCCTCCAGGTCCCCACGGCCC
IE	CCTGATCTGGGTCTCCCTCCCACCCTCAGGGAGGCCAGGCTCGG
IE	AGCCCTCAACCCTTCTGTCTCACCCCTCCAGCCTAAAGCTCCTGA
IE	CCACTCAGCCAGGCCCTTCTTCCTCCAGGTCCCCACGGCCC
N	CTGTGTTACCAACATCAAGCGCCGGACATCGTGCTCAAGTGGG
N	GTGTTACCGAGGGCATTCTAACAGTCTTACTACGGCCTCCG
N	TCTGAGCTCTGCATTGTCTATTCTCCAGCTGACCCTGGTTCTCT

Cross-discipline Capability



A	we
T	you
C	he
G	she

DNA sequence



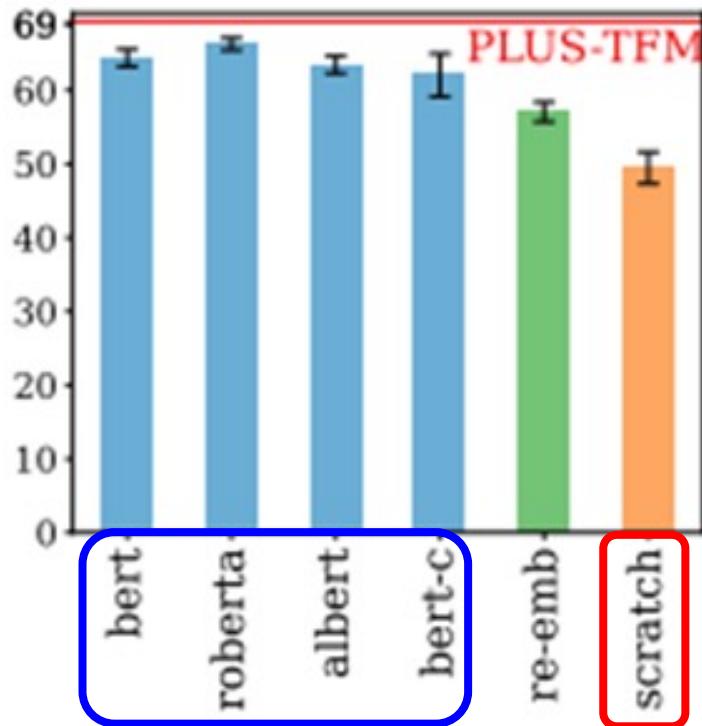
Cross-discipline Capability

- Applying BERT to **protein, DNA, music classification**

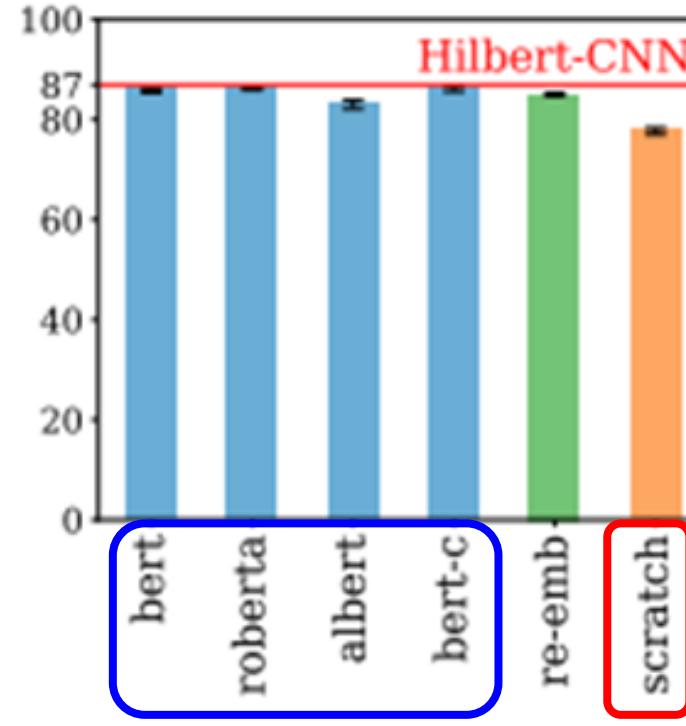
	Protein			DNA			Music	
	localization	stability	fluorescence	H3	H4	H3K9ac	Splice	composer
specific	69.0	76.0	63.0	87.3	87.3	79.1	94.1	-
BERT	64.8	74.5	63.7	83.0	86.2	78.3	97.5	55.2
re-emb	63.3	75.4	37.3	78.5	83.7	76.3	95.6	55.2
rand	58.6	65.8	27.5	75.6	66.5	72.8	95	36

The pretrained models learn some general skills for the classification.

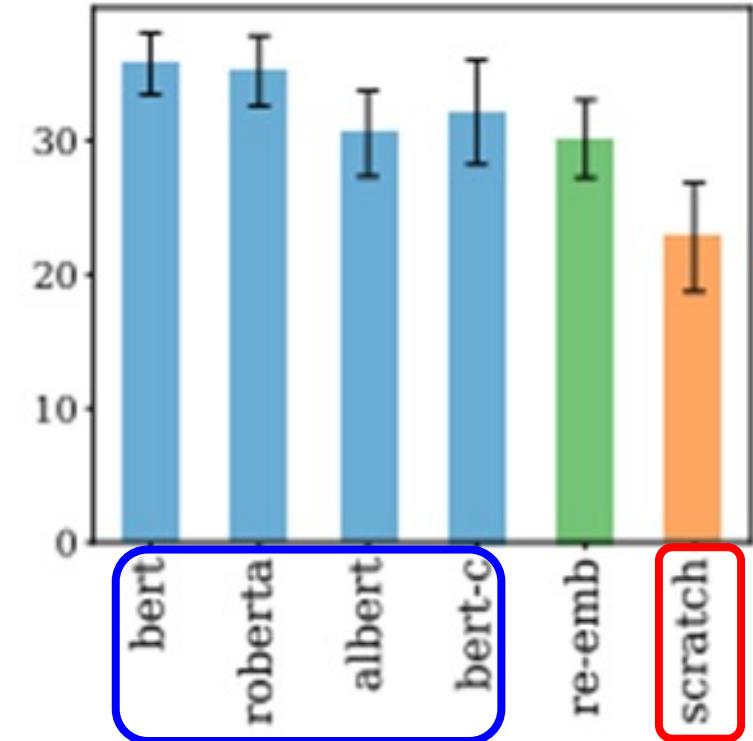
Cross-discipline Capability



DNA Classification
(average 3 tasks)



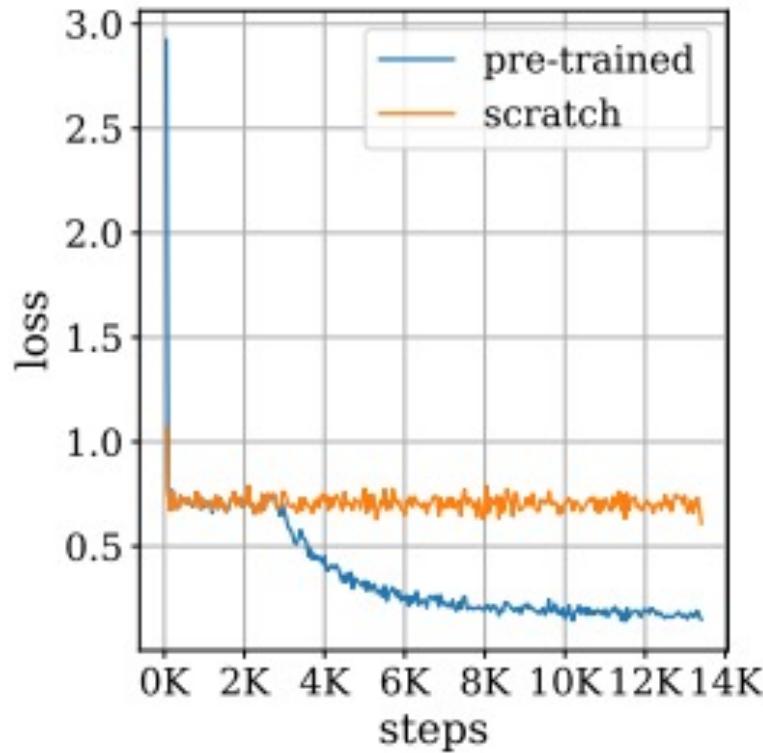
Protein Classification
(average 4 tasks)



Music Classification

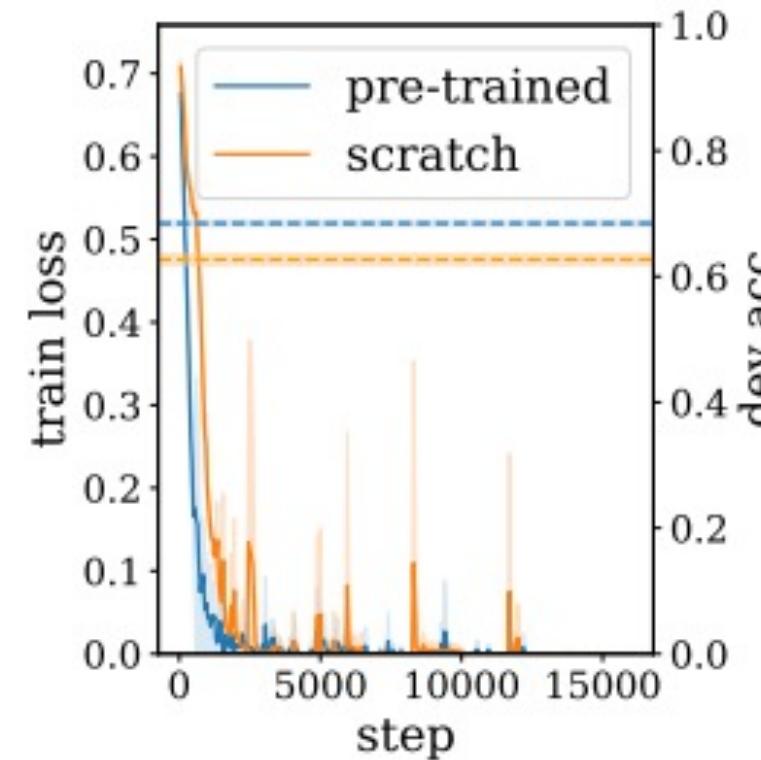
How pre-trained model improve the performance?

Optimization

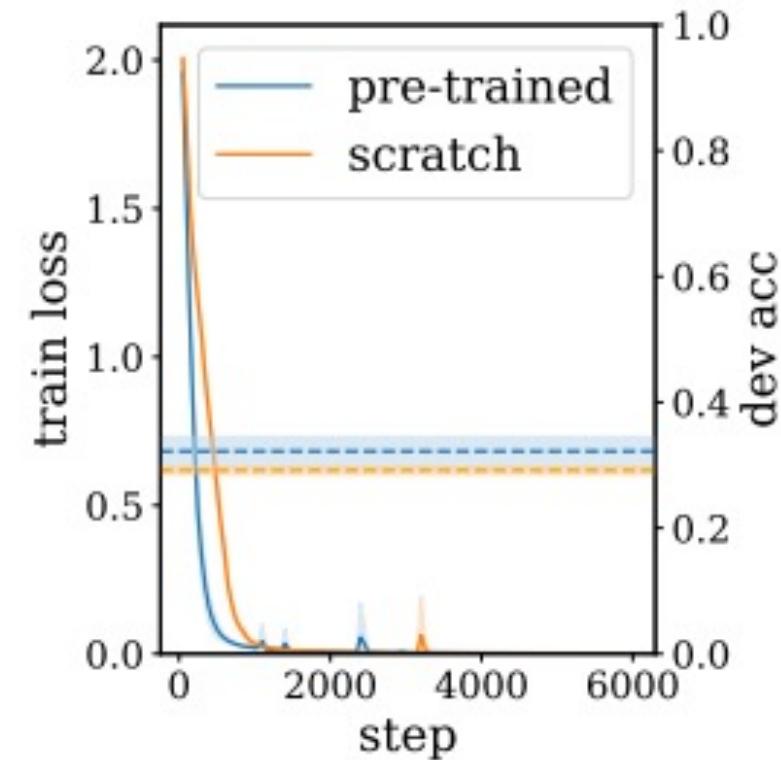


(b) fluorescence

Generalization



(a) H3K9ac

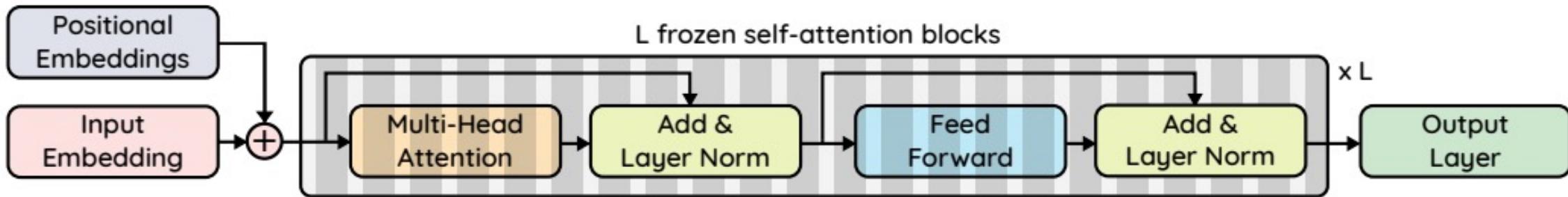


(b) localization

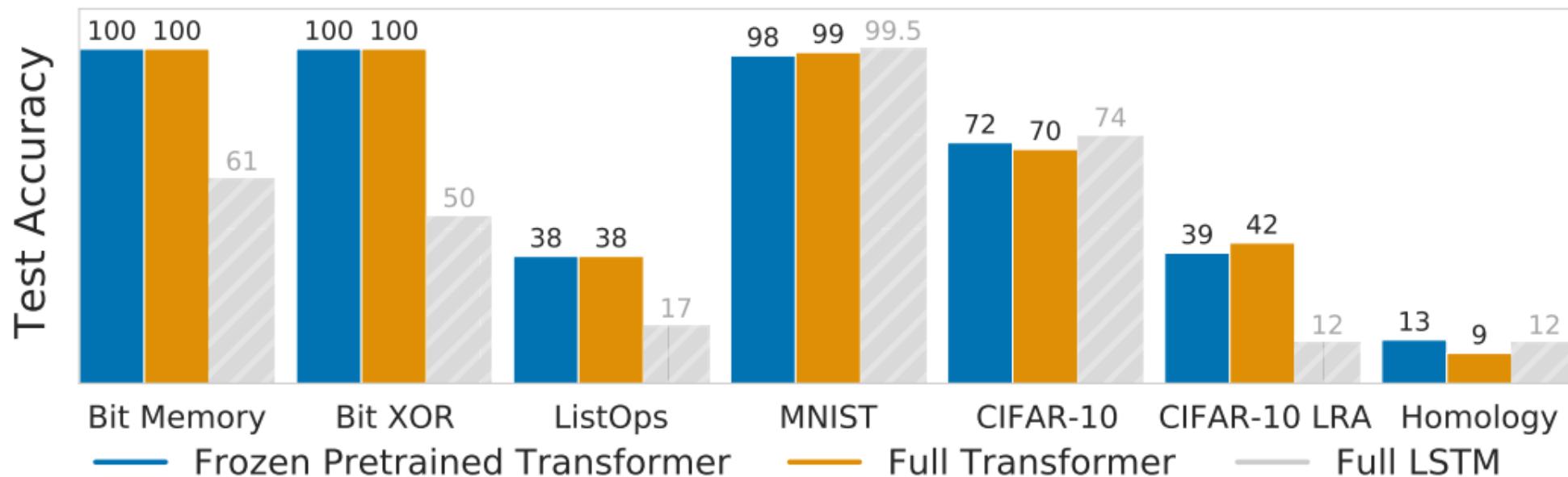
The pretrained models help both optimization and generalization.

To learn more ...

Kevin Lu, Aditya Grover, Pieter Abbeel, Igor Mordatch, Pretrained Transformers as Universal Computation Engines, arXiv, 2021



Performance on Multimodal Sequence Benchmarks

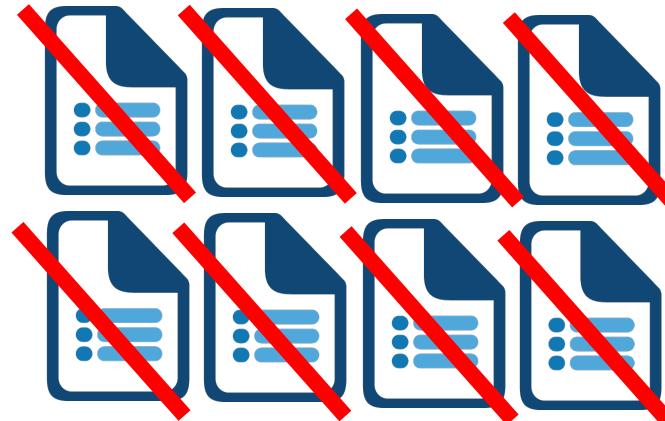


Pre-training on Artificial Data

Cheng-Han Chiang, Hung-yi Lee, On the Transferability of Pre-trained Language Models: A Study from Artificial Datasets, AAAI, 2022

We have seen the cross-discipline capability of self-supervised model

Pre-train



We can pre-train model on
data other than text.

Fine-tune



A set of NLP Tasks

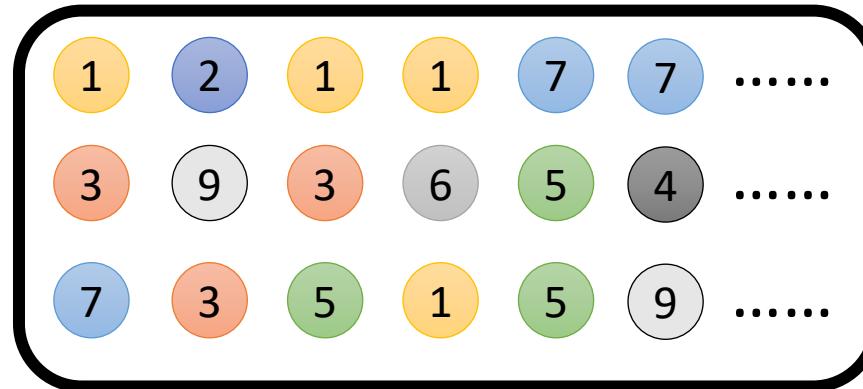
Testing

Pre-training on Artificial Data

Cheng-Han Chiang, Hung-yi Lee, On the Transferability of Pre-trained Language Models: A Study from Artificial Datasets, AAAI, 2022

We have seen the cross-discipline capability of self-supervised model

Pre-train



Token generation by rules

Fine-tune

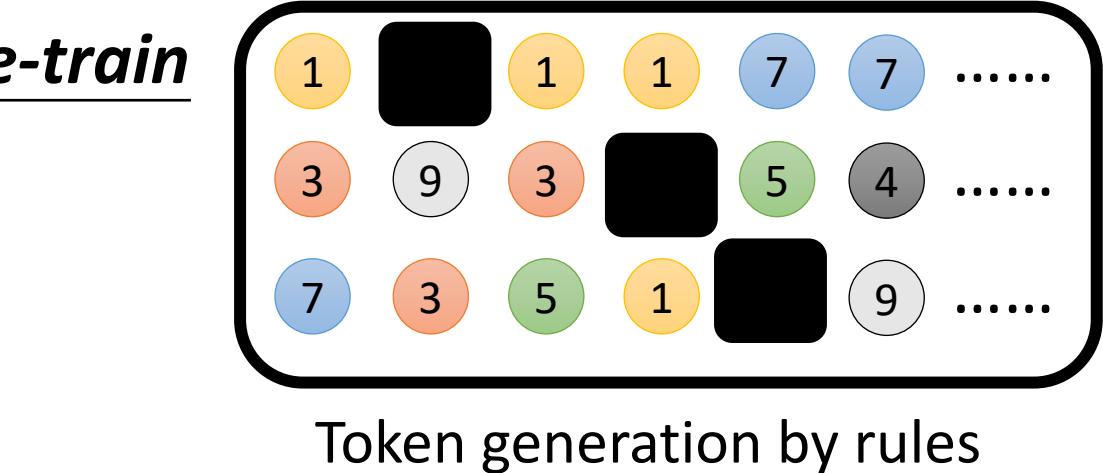


A set of NLP Tasks

Testing

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

Pre-train



NLP Downstream Task: Sentiment Analysis

pos/neg

Linear

Fine-tune

BERT pre-training on token IDs
(take token IDs as input)

[CLS]

1

101

3

4

the

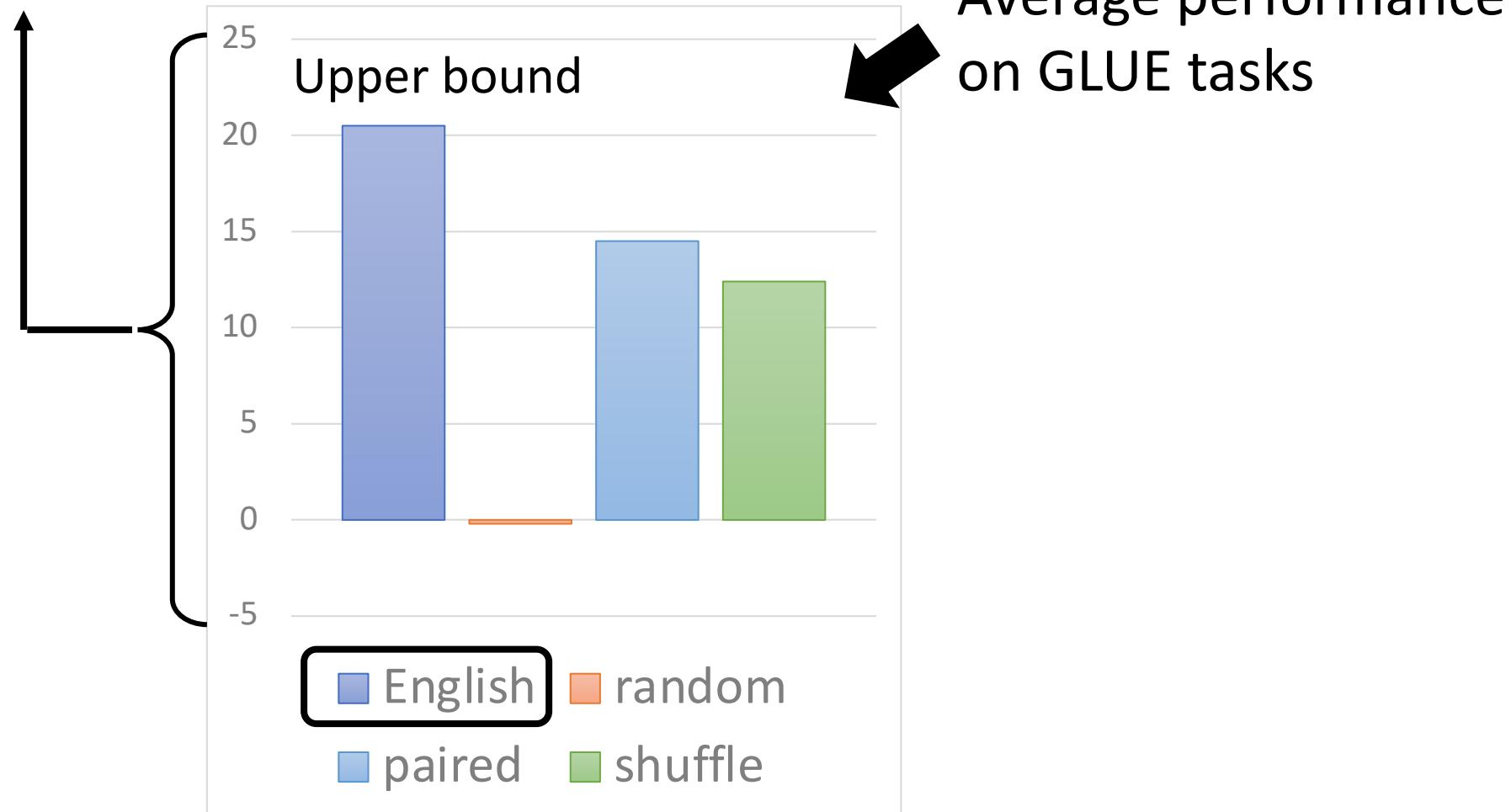
movie

is

good

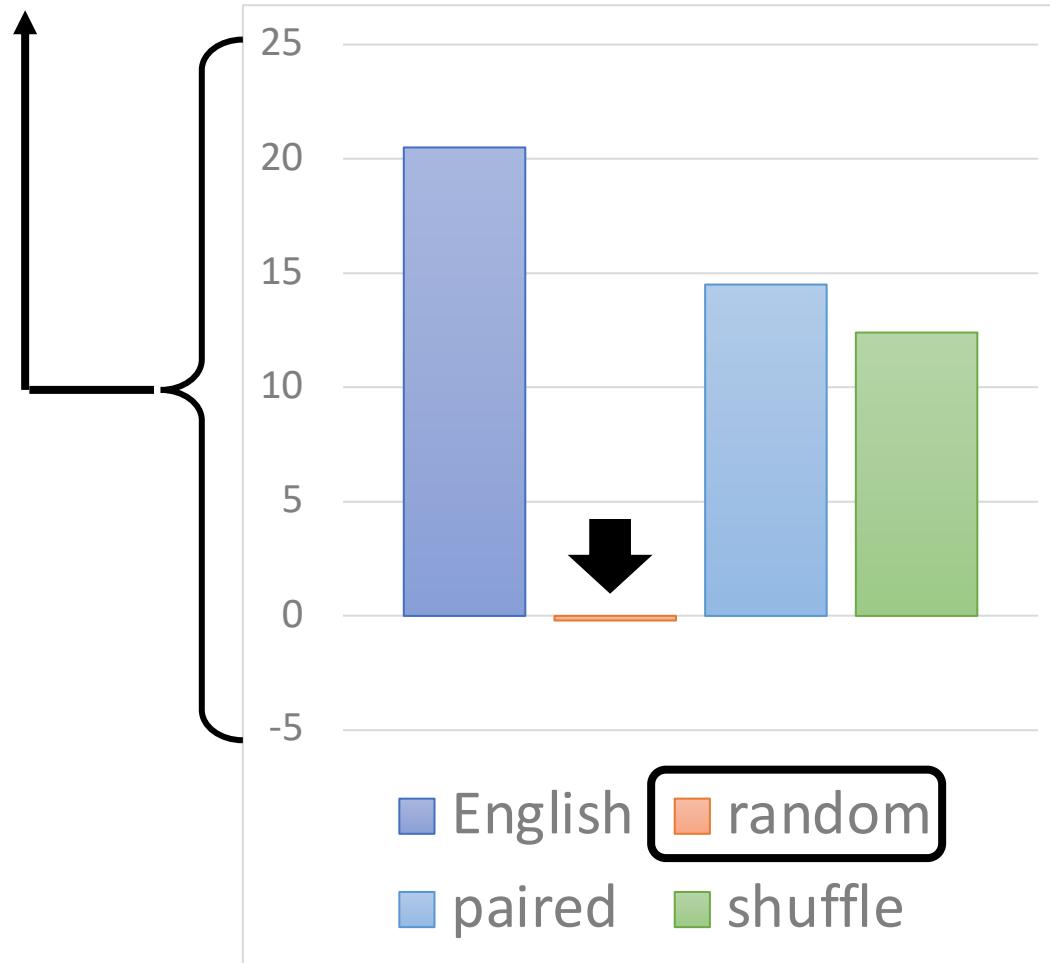
Pre-training on Artificial Data

Absolute improvement (%) compared
to training from scratch



Pre-training on Artificial Data

Absolute improvement (%) compared to training from scratch

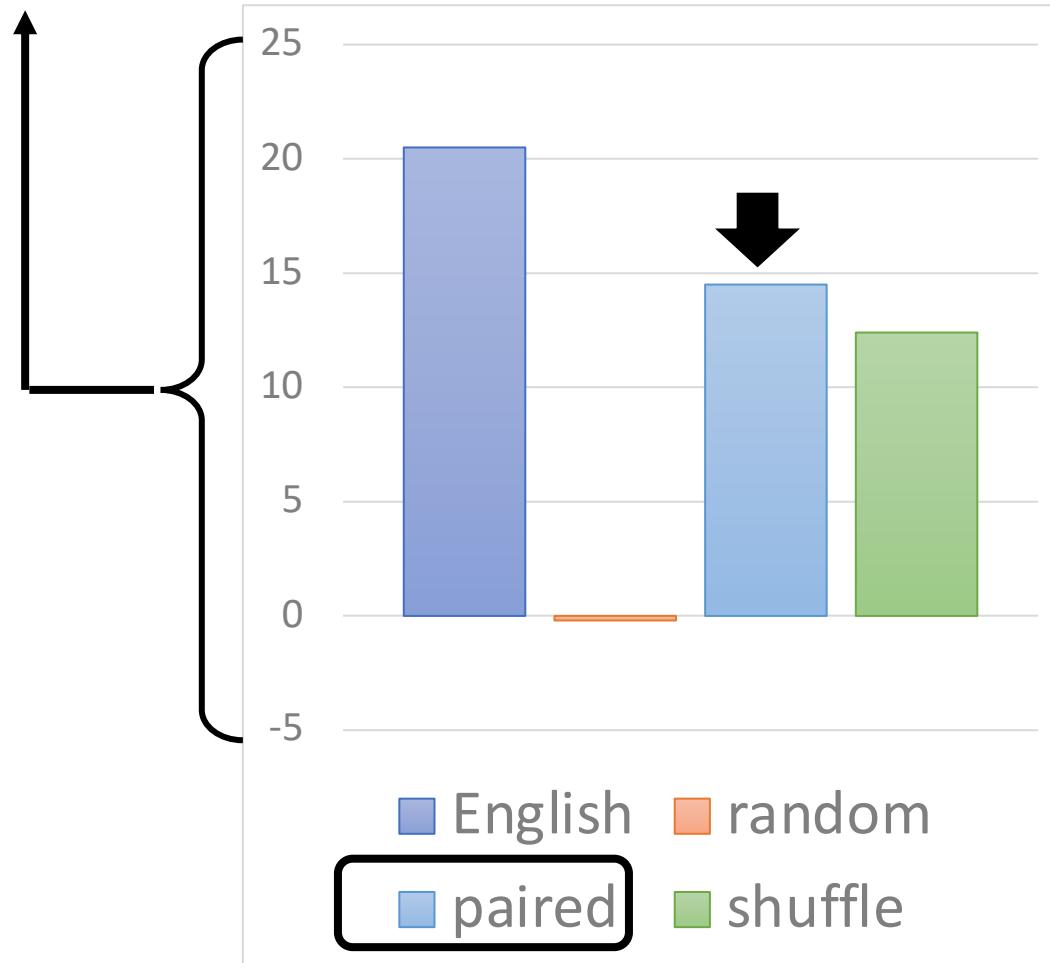


- Pre-training on random tokens yields the same performance as training from scratch.

Data plays the role.

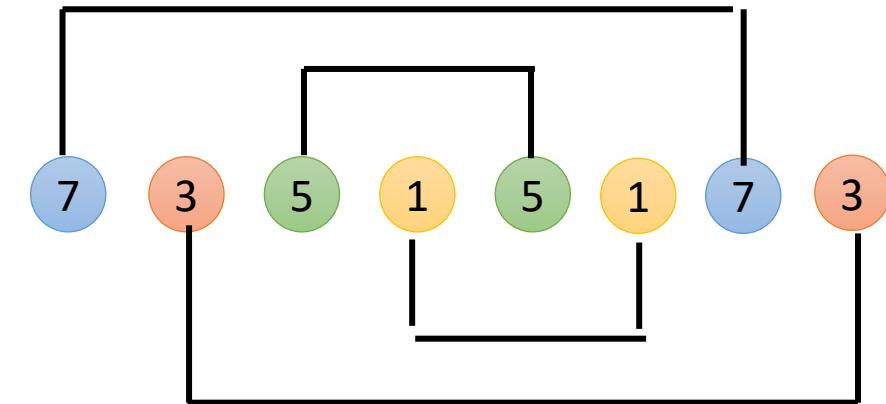
Pre-training on Artificial Data

Absolute improvement (%) compared to training from scratch



Also refer to:
Isabel Papadimitriou, Dan Jurafsky,
Learning Music Helps You Read: Using
Transfer to Study Linguistic Structure
in Language Models, EMNLP, 2020

- All the tokens in the generated sequences are paired.

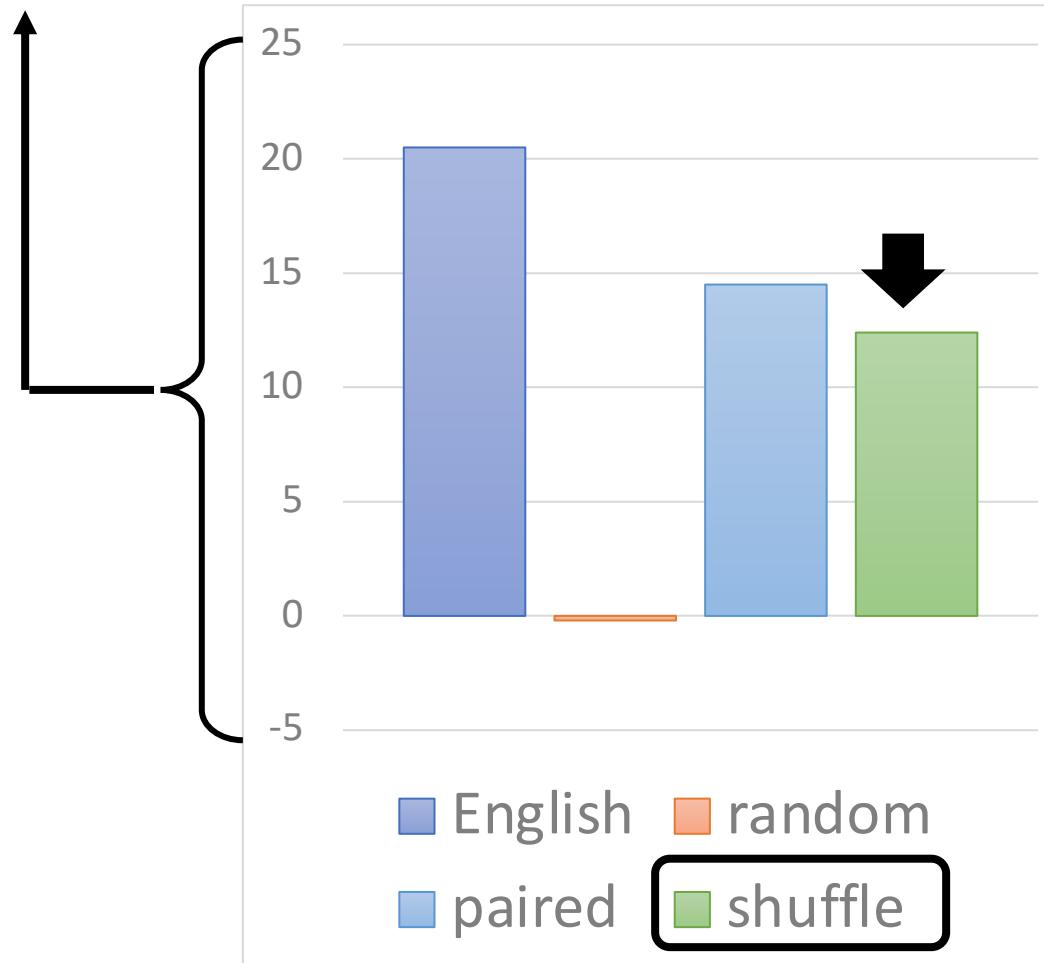


Structured data is critical for learning useful skills for NLP.

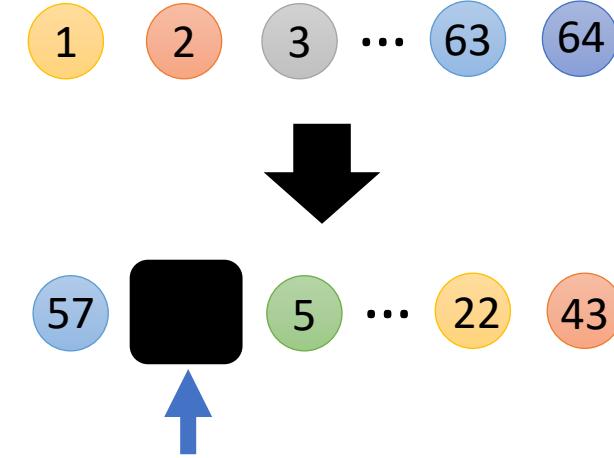
Is it true?

Pre-training on Artificial Data

Absolute improvement (%) compared to training from scratch



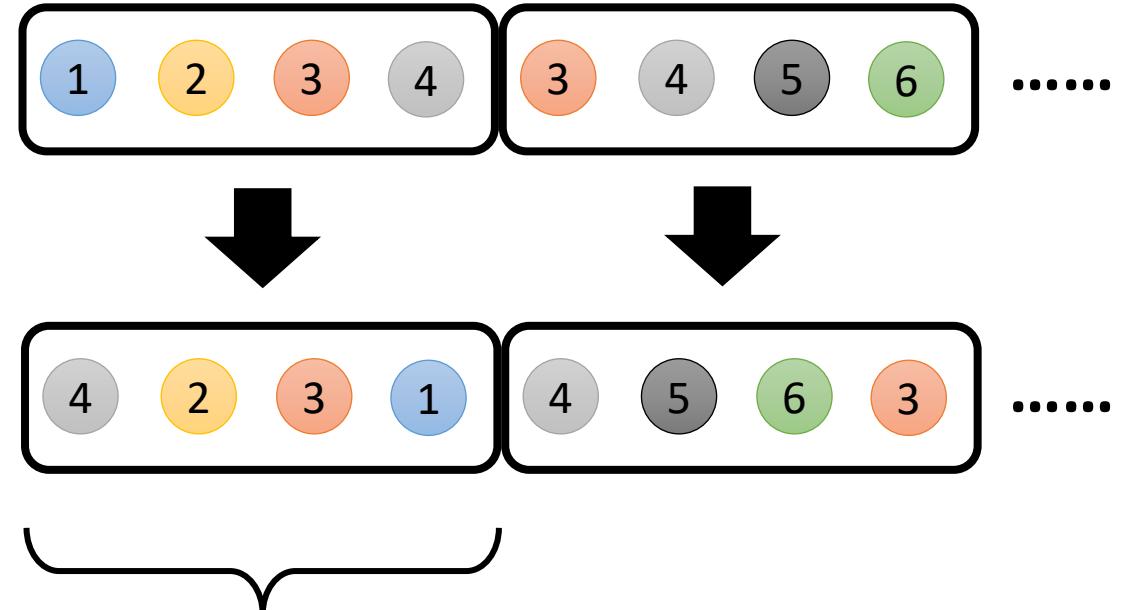
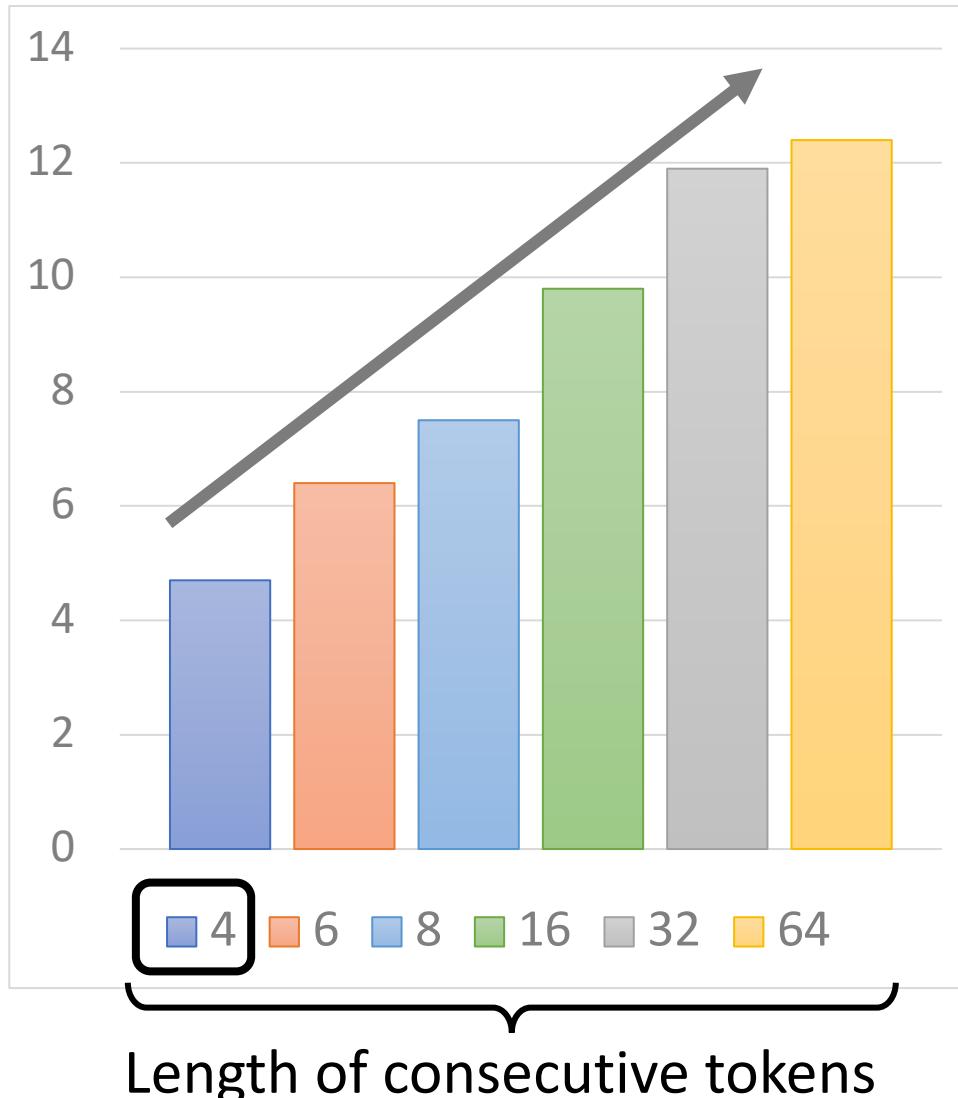
- Shuffle



To predict this token, model needs to go through the whole sequence.

Is long-range reading the key to the success of a pretrained model?

Absolute improvement (%) compared to training from scratch



Longer consecutive tokens, better performance in NLP tasks

Learning to read a long-range in a sequence is crucial.

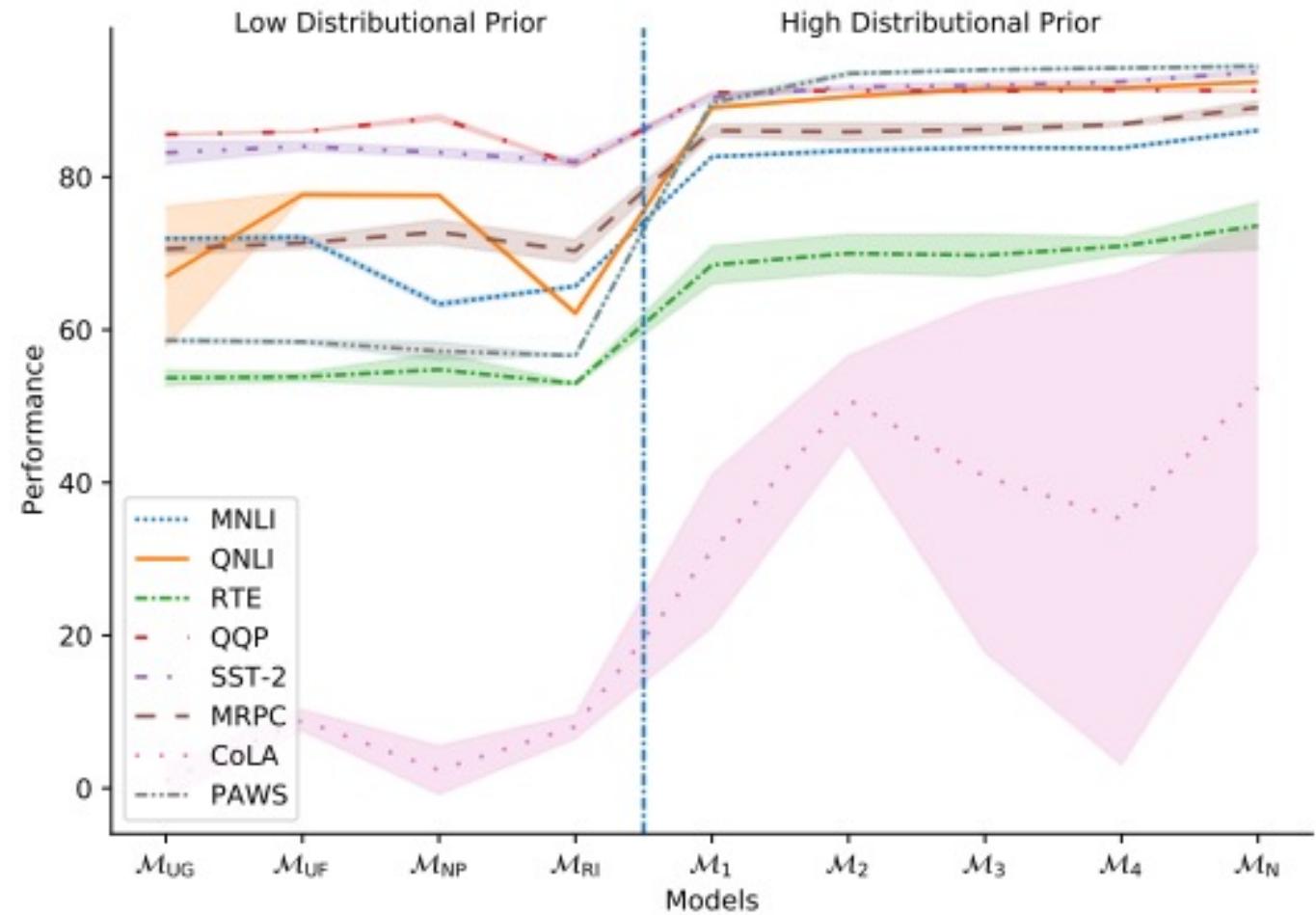
Are there more factors?
Need more investigation 😊

To learn more

AACL-IJCNLP 2022 will be held online from November 20-23 , 2022 .

will 2022 , be November . online 2022 from held 20-23 AACL-IJCNLP

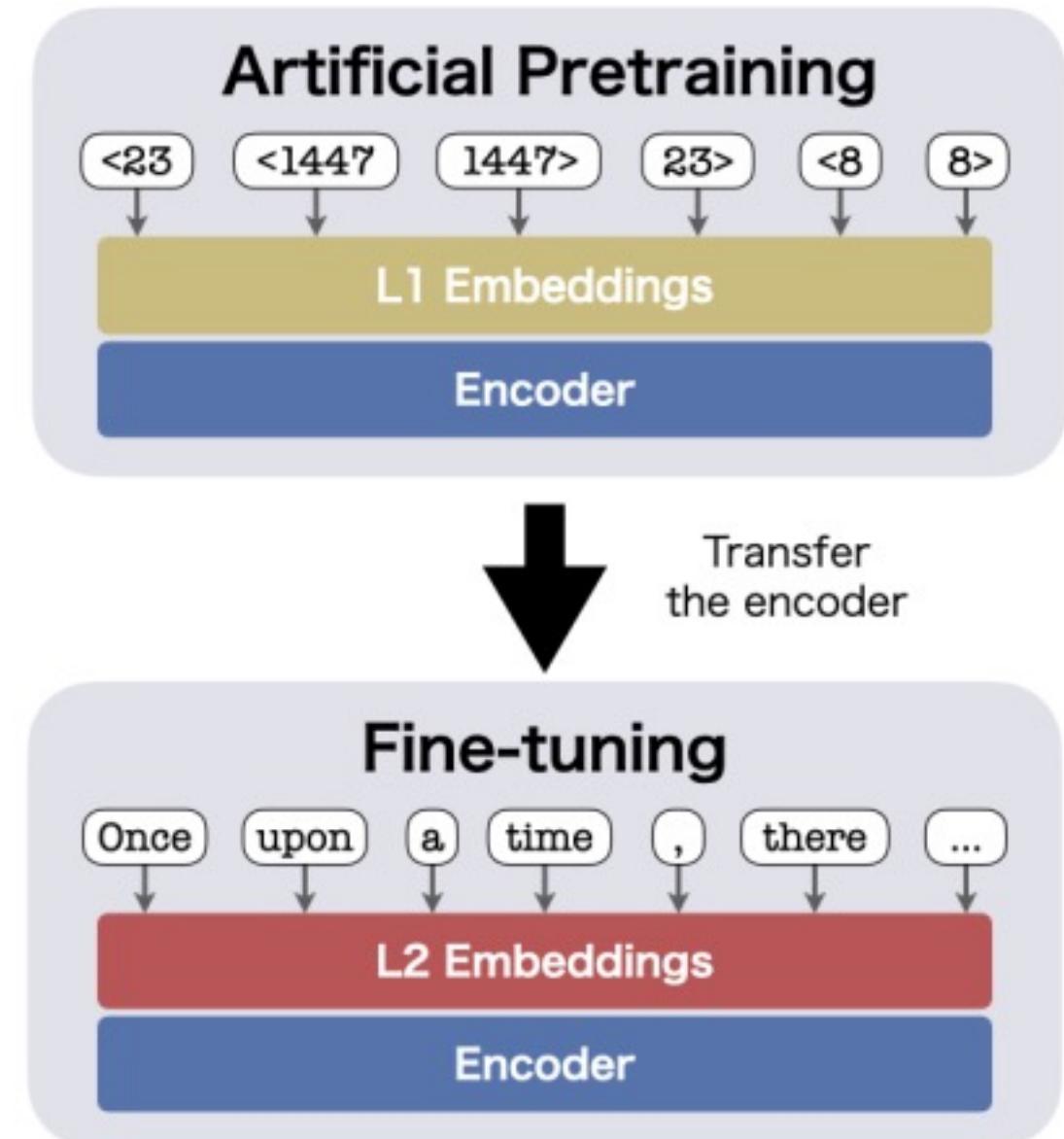
Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, Douwe Kiela, Masked Language Modeling and the Distributional Hypothesis: Order Word Matters Pre-training for Little, EMNLP, 2021



To learn more

- What knowledge in pretrained encoders are transferred across different languages?

“tokens in a sequence can be characterized by its neighbor tokens at specific positions”



Ryokan Ri, Yoshimasa Tsuruoka, Pretraining with Artificial Language: Studying Transferable Knowledge in Language Models, ACL, 2022

Concluding Remarks of Part 2

- Why do PLMs work?

Don't know the answer yet.

- Contextualized word representations
- BERTology: Analyzing what is learned by BERT
- BERT Embryology: Analyzing what BERT learned during training
- Cross-discipline Capability
- Pre-training on Artificial Data

Schedule

- 17:00 – 17:10 **Part 1** Introduction [Hung-yi]
- 17:10 – 17:40 **Part 2** Why do PLMs work [Hung-yi]
- 17:40 – 18:20 **Part 3** How to use PLMs: Contrastive Learning for PLMs [Yung-Sung]
- 18:20 – 18:30 Q&A for Part 1+2+3
- 18:30 – 18:40 Break
- 18:40 – 19:05 **Part 4** How to use PLMs: Parameter-efficient fine-tuning [Cheng-Han]
- 19:05 – 19:50 **Part 5** How to use PLMs: Using PLMs with different amounts of data [Cheng-Han]
- 19:50 – 20:00 Conclusion and Future work + Q&A

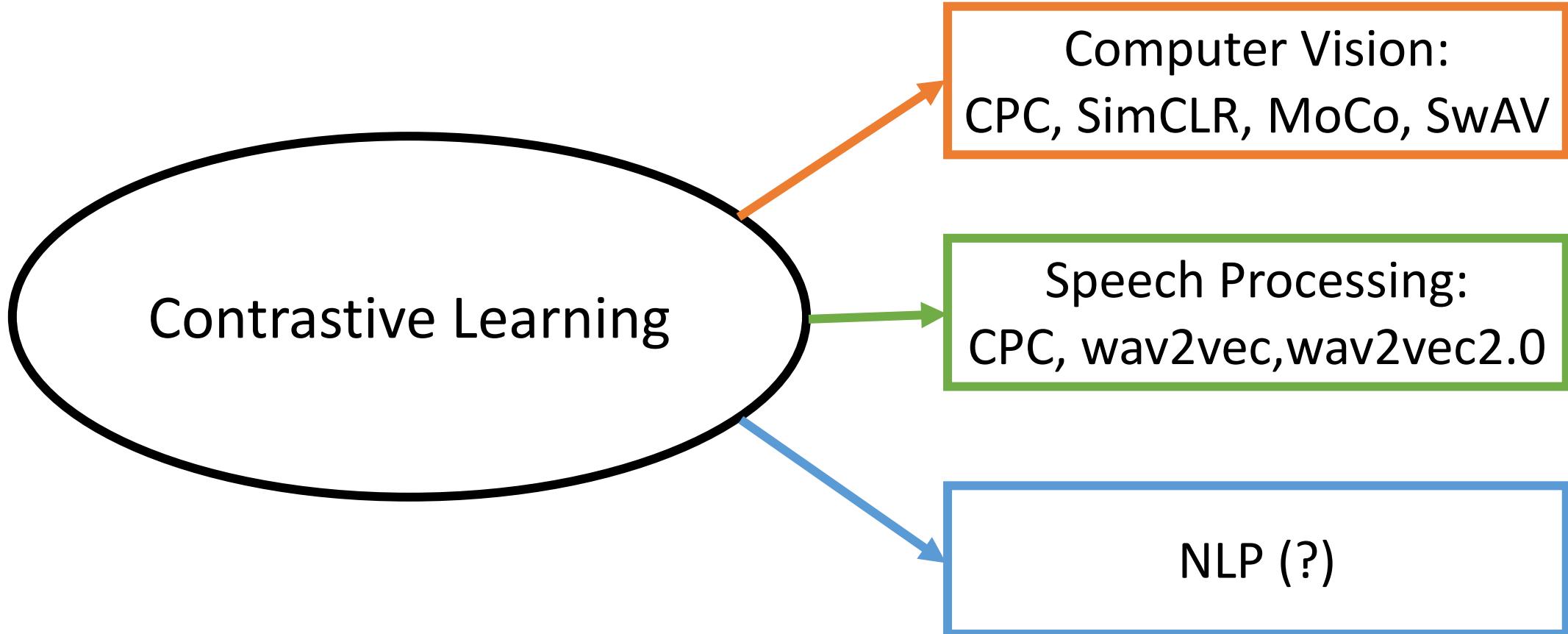


2022 AACL-IJCNLP

Part 3: How to use PLMs: Contrastive Learning for PLMs

Yung-Sung Chuang
CSAIL, MIT





Why Contrastive?

We want to obtain a good representation space such that

1. Similar inputs have similar representations. -> **Positive Pairs**



Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *International conference on machine learning*. PMLR, 2020.

Khosla, Prannay, et al. "Supervised contrastive learning." *Advances in Neural Information Processing Systems* 33 (2020): 18661-18673.

Why Contrastive?

We want to obtain a good representation space such that

2. Dissimilar inputs have dissimilar representations. -> **Negative Pairs**



Randomly sampled images

Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *International conference on machine learning*. PMLR, 2020.

Khosla, Prannay, et al. "Supervised contrastive learning." *Advances in Neural Information Processing Systems* 33 (2020): 18661-18673.

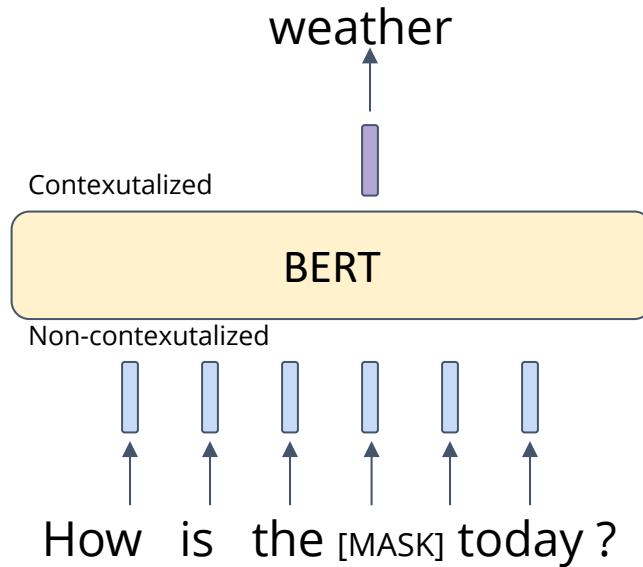
Contrastive Learning

SimCLR for Computer Vision

$$\ell_i = \log \frac{e^{\text{sim}(\mathbf{h}_i, \mathbf{h}_i^+)/\tau}}{\sum_{j=1}^N e^{\text{sim}(\mathbf{h}_i, \mathbf{h}_j^+)/\tau}}$$

Contrastive Learning for NLP?

- Masked Language Modeling shares some similarity to contrastive learning



Instance:

contextualized representation of [MASK]

Positive Pairs:

non-contextualized representation of "weather"

Negative Pairs:

non-contextualized representation of all the other words in the vocabulary

Why Contrastive on NLP?

- MLM can be seen as a contrastive learning task using all negative pairs for training
- Finite vocabulary size (30k for BERT) prevents negative sampling issues
 - > MLM can be trained as a simple token-level classification task
- Are there any task has infinite possible inputs?

→ **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.

Outline of Part 3

1. Why we need sentence-level representations?
2. Pre-BERT methods
3. How to obtain sentence-level representations from BERTs?
 - a. Post-processing Methods
4. Contrastive Learning Methods:
 - a. Designed Positives
 - b. Generating Positives
 - c. Bootstrapping Methods
 - d. Dropout Augmentations
 - e. Equivariant Contrastive Learning
 - f. Prompting
 - g. Ranking-based Methods
5. Conclusion

Outline of Part 3

1. **Why we need sentence-level representations?**
2. Pre-BERT methods
3. How to obtain sentence-level representations from BERTs?
 - a. Post-processing Methods
4. Contrastive Learning Methods:
 - a. Designed Positives
 - b. Generating Positives
 - c. Bootstrapping Methods
 - d. Dropout Augmentations
 - e. Equivariant Contrastive Learning
 - f. Prompting
 - g. Ranking-based Methods
5. Conclusion

General-Purpose Sentence 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

We will mainly focus on unsupervised methods through this tutorial!

Before BERT came out...

- Skip-Thought Vectors, NIPS 2016 -> *Next Sentence Prediction*

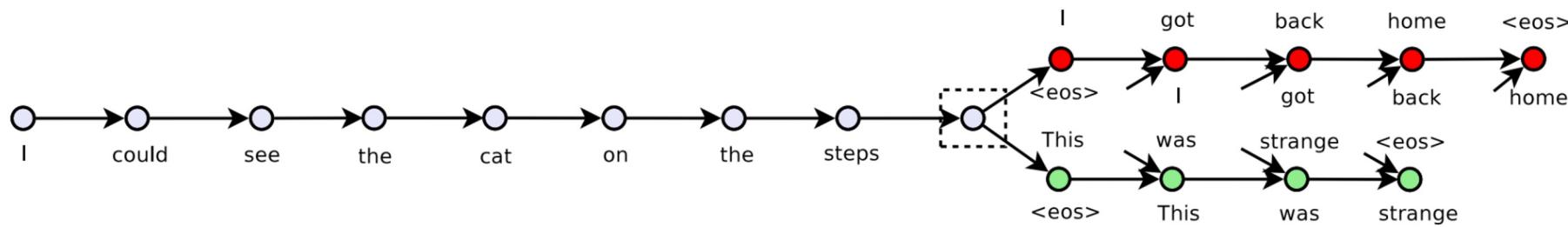
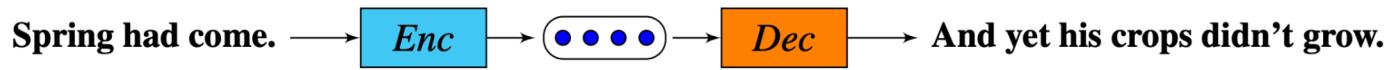


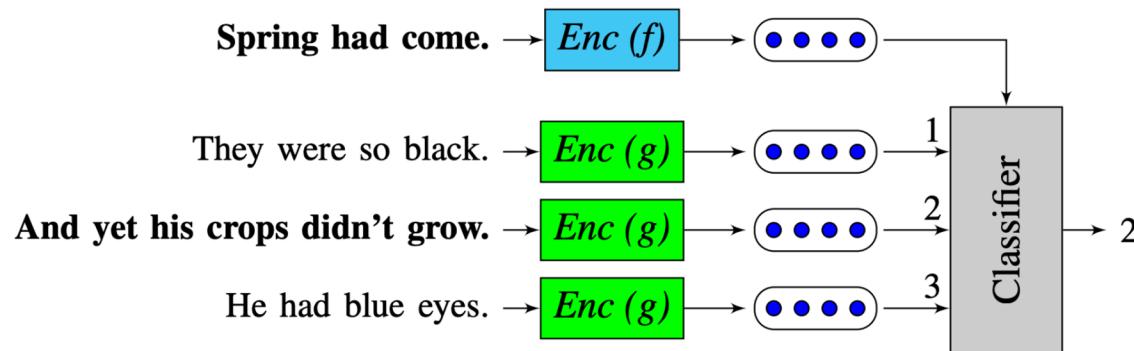
Figure 1: The skip-thoughts model. Given a tuple (s_{i-1}, s_i, s_{i+1}) of contiguous sentences, with s_i the i -th sentence of a book, the sentence s_i is encoded and tries to reconstruct the previous sentence s_{i-1} and next sentence s_{i+1} . In this example, the input is the sentence triplet *I got back home. I could see the cat on the steps. This was strange.* Unattached arrows are connected to the encoder output. Colors indicate which components share parameters. $\langle \text{eos} \rangle$ is the end of sentence token.

Before BERT came out...

- Quick-Thought vectors, ICLR 2018 -> *Next Sentence Prediction w/o Decoder*



(a) Conventional approach



(b) Proposed approach

Oct 2018:

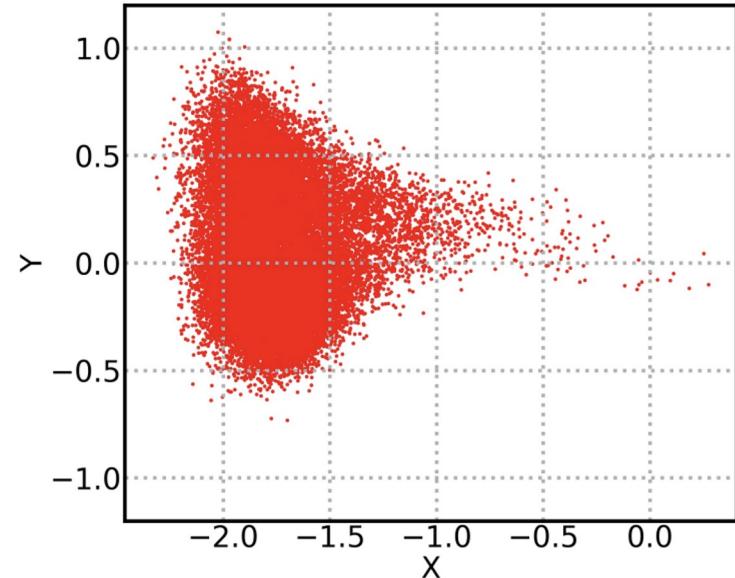


How to obtain sentence representations from BERT?

- It cannot be trivially obtained from token-level representations
- Average pooling performs even worse than avg. GloVe embeddings

Dataset	STS-B	SICK-R	STS-12	STS-13	STS-14	STS-15	STS-16
<i>Published in (Reimers and Gurevych, 2019)</i>							
Avg. GloVe embeddings	58.02	53.76	55.14	70.66	59.73	68.25	63.66
Avg. BERT embeddings	46.35	58.40	38.78	57.98	57.98	63.15	61.06
BERT CLS-vector	16.50	42.63	20.16	30.01	20.09	36.88	38.03

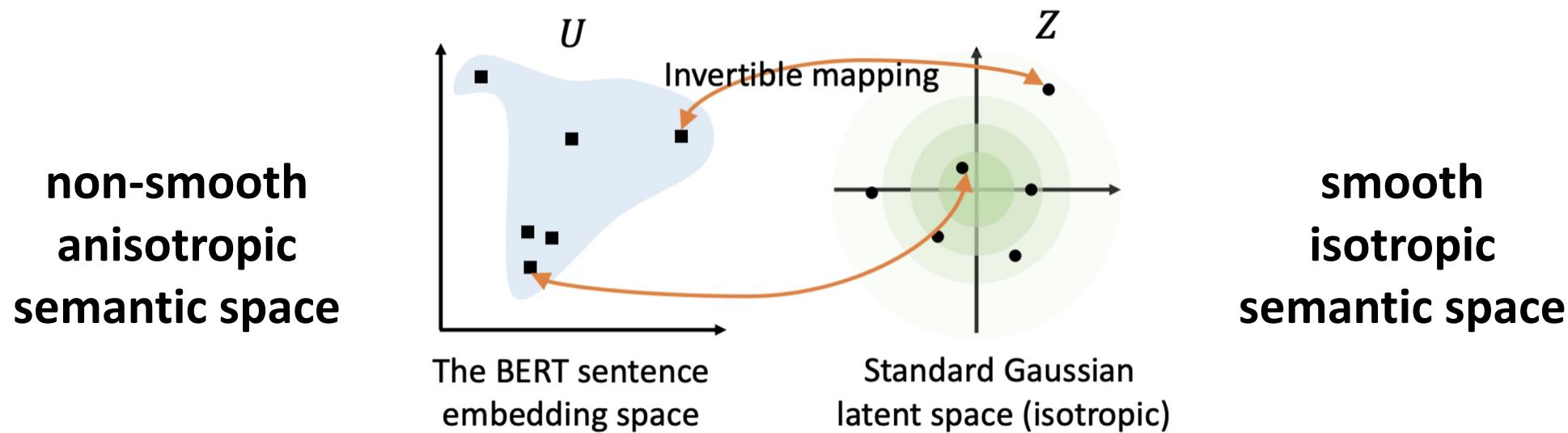
Anisotropy problem in BERT's representation space



- **Representation degeneration:** the learned embeddings occupy a narrow cone in the vector space
- Limits the expressiveness of the vector space.

Gao, Jun, et al. "Representation Degeneration Problem in Training Natural Language Generation Models." *International Conference on Learning Representations*. 2018.
Ethayarajh, Kawin. "How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019.
Wang, Lingxiao, et al. "Improving neural language generation with spectrum control." *International Conference on Learning Representations*. 2019.

BERT-flow



BERT-flow

- **Sentence Textual Similarity (STS) Task**

Data: Sentence pairs with 1-5 human ratings for the similarity

Metric: Spearman Correlation between model predictions and human ratings

Dataset	STS-B	SICK-R	STS-12	STS-13	STS-14	STS-15	STS-16
<i>Published in (Reimers and Gurevych, 2019)</i>							
Avg. GloVe embeddings	58.02	53.76	55.14	70.66	59.73	68.25	63.66
Avg. BERT embeddings	46.35	58.40	38.78	57.98	57.98	63.15	61.06
BERT CLS-vector	16.50	42.63	20.16	30.01	20.09	36.88	38.03
<i>Our Implementation</i>							
BERT _{base}	47.29	58.21	49.07	55.92	54.75	62.75	65.19
BERT _{base} -last2avg	59.04	63.75	57.84	61.95	62.48	70.95	69.81
BERT _{base} -flow (NLI*)	58.56 (↓)	65.44 (↑)	59.54 (↑)	64.69 (↑)	64.66 (↑)	72.92 (↑)	71.84 (↑)
BERT _{base} -flow (target)	70.72 (↑)	63.11(↓)	63.48 (↑)	72.14 (↑)	68.42 (↑)	73.77 (↑)	75.37 (↑)

BERT-whitening

- Using a simple whitening post-processing can outperform BERT-flow

	STS-B	STS-12	STS-13	STS-14	STS-15	STS-16	SICK-R
<i>Published in (Reimers and Gurevych, 2019)</i>							
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
<i>Published in (Li et al., 2020)</i>							
BERT _{base} -first-last-avg	59.04	57.84	61.95	62.48	70.95	69.81	63.75
BERT _{base} -flow (NLI)	58.56	59.54	64.69	64.66	72.92	71.84	65.44
BERT _{base} -flow (target)	70.72	63.48	72.14	68.42	73.77	75.37	63.11
<i>Our implementation</i>							
BERT _{base} -first-last-avg	59.04	57.86	61.97	62.49	70.96	69.76	63.75
BERT _{base} -whitening (NLI)	68.19(↑)	61.69(↑)	65.70(↑)	66.02(↑)	75.11(↑)	73.11(↑)	63.6(↓)
BERT _{base} -whitening-256 (NLI)	67.51(↑)	61.46(↑)	66.71(↑)	66.17(↑)	74.82(↑)	72.10(↑)	64.9(↓)
BERT _{base} -whitening (target)	71.34(↑)	63.62(↑)	73.02(↑)	69.23(↑)	74.52(↑)	72.15(↓)	60.6(↓)
BERT _{base} -whitening-256 (target)	71.43(↑)	63.89(↑)	73.76(↑)	69.08(↑)	74.59(↑)	74.40(↓)	62.2(↓)

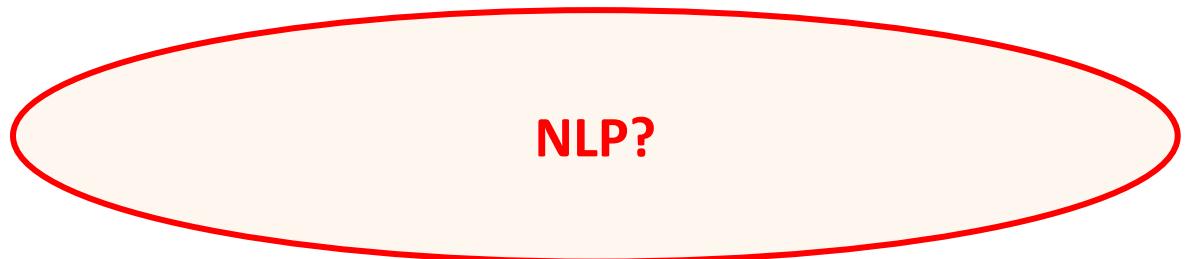
We need further fine-tuning
to extract better sentence embeddings
from pre-trained language models...

Outline of Part 3

1. Why we need sentence-level representations?
2. Pre-BERT methods
3. How to obtain sentence-level representations from BERTs?
 - a. Post-processing Methods
4. **Contrastive Learning Methods:**
 - a. Designed Positives
 - b. Generating Positives
 - c. Bootstrapping Methods
 - d. Dropout Augmentations
 - e. Equivariant Contrastive Learning
 - f. Prompting
 - g. Ranking-based Methods
5. Conclusion

Contrastive Learning

How can we produce
augmentations in NLP?



NLP?

Outline of Part 3

1. Why we need sentence-level representations?
2. Pre-BERT methods
3. How to obtain sentence-level representations from BERTs?
 - a. Post-processing Methods
4. Contrastive Learning Methods:
 - a. **Designed Positives**
 - b. Generating Positives
 - c. Bootstrapping Methods
 - d. Dropout Augmentations
 - e. Equivariant Contrastive Learning
 - f. Prompting
 - g. Ranking-based Methods
5. Conclusion

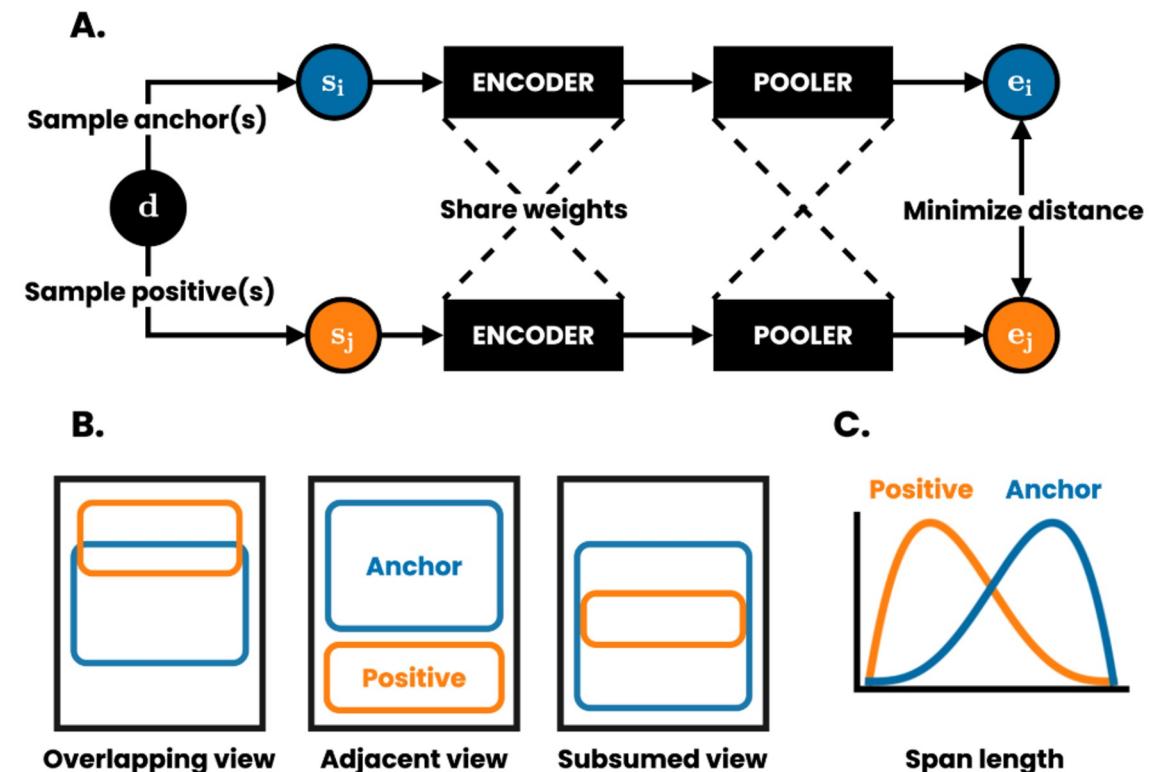
DeCLUTR

- **Positive Pairs:**

Overlapping/adjacent spans from the same document

- **Negative Pairs:**

- **hard** negatives from the same docs
- **easy** negatives from different docs



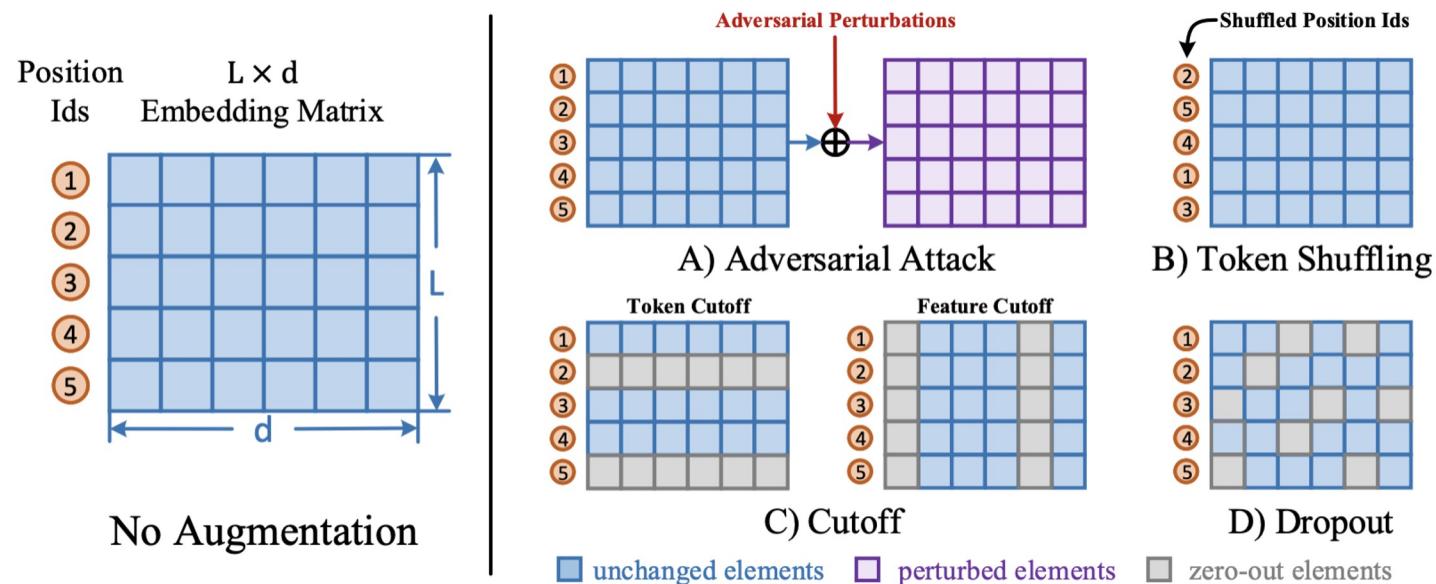
Results (STS)

- Good improvement over post-processing methods

Model	SICK-E	SICK-R	STS-B	COCO	STS12*	STS13*	STS14*	STS15*	STS16*
GloVe	78.89	72.30	62.86	0.40	53.44	51.24	55.71	59.62	57.93
fastText	79.01	72.98	68.26	0.40	58.85	58.83	63.42	69.05	68.24
InferSent	86.30	83.06	78.48	65.84	62.90	56.08	66.36	74.01	72.89
USE	85.37	81.53	81.50	62.42	68.87	71.70	72.76	83.88	82.78
Sent. Transformers	82.97	79.17	74.28	60.96	64.10	65.63	69.80	74.71	72.85
QuickThoughts	—	—	—	60.55	—	—	—	—	—
Transformer-small	81.96	77.51	70.31	60.48	53.99	45.53	57.23	65.57	63.51
Transformer-base	80.29	76.84	69.62	60.14	53.28	46.10	56.17	64.69	62.79
DeCLUTR-small	83.46 ↑	77.66 ↑	77.51 ↑	60.85 ↑	63.66 ↑	68.93 ↑	70.40 ↑	78.25 ↑	77.74 ↑
DeCLUTR-base	83.84 ↑	78.62 ↑	79.39 ↑	62.35 ↑	63.56 ↑	72.58 ↑	71.70 ↑	79.95 ↑	79.59 ↑
BERT-flow	--	63.11	70.72	--	63.48	72.02	68.42	73.77	75.37
BERT-whitening	--	62.20	71.43	--	63.89	73.76	69.08	74.59	74.40

ConSERT

- All the possible augmentations on token embedding space

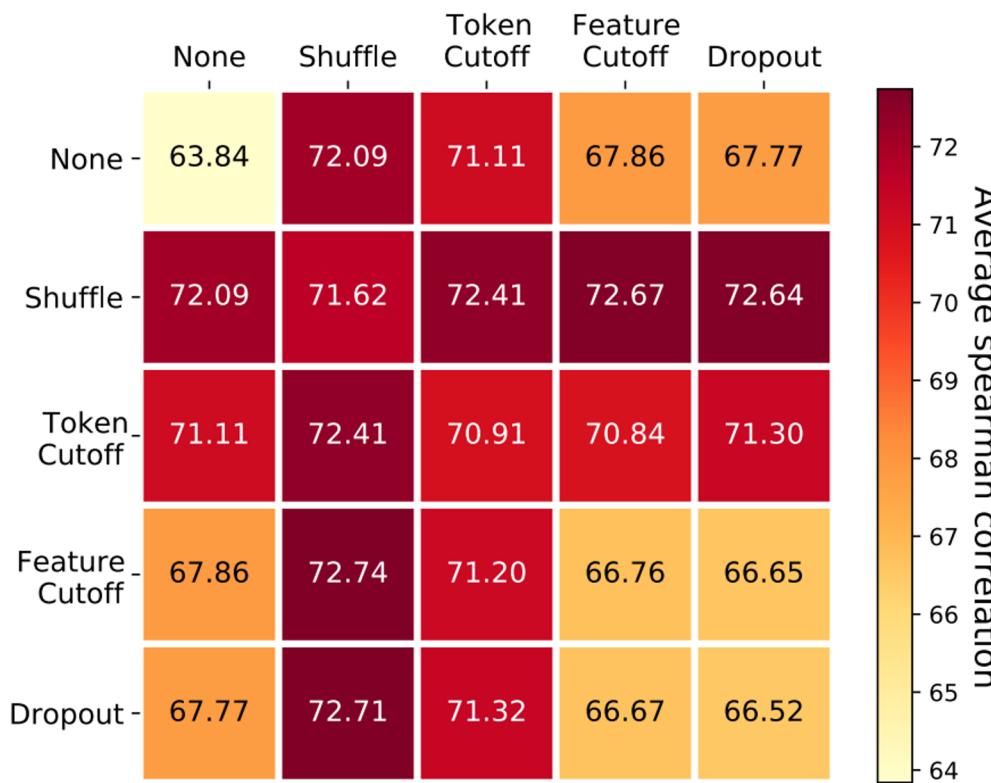


Experiments

- Using unlabeled text to train contrastive loss for adaptation

Method	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
<i>Unsupervised baselines</i>								
Avg. GloVe embeddings [†]	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
BERT _{base} [‡]	35.20	59.53	49.37	63.39	62.73	48.18	58.60	53.86
BERT _{large} [‡]	33.06	57.64	47.95	55.83	62.42	49.66	53.87	51.49
CLEAR _{base} [†]	49.0	48.9	57.4	63.6	65.6	75.6	72.5	61.8
IS-BERT _{base} -NLI [†]	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
BERT _{base} -CT [†]	66.86	70.91	72.37	78.55	77.78	-	-	-
BERT _{large} -CT [†]	69.50	75.97	74.22	78.83	78.92	-	-	-
<i>Using STS unlabeled texts</i>								
BERT _{base} -flow [†]	63.48	72.14	68.42	73.77	75.37	70.72	63.11	69.57
BERT _{large} -flow [†]	65.20	73.39	69.42	74.92	77.63	72.26	62.50	70.76
ConSERT _{base} [‡]	64.64	78.49	69.07	79.72	75.95	73.97	67.31	72.74
ConSERT _{large} [‡]	70.69	82.96	74.13	82.78	76.66	77.53	70.37	76.45
DeCLUTR (BERT-base)	63.56	72.58	71.70	79.95	79.59	79.39	78.62	75.06

Effects of Augmentation Strategies



Outline of Part 3

1. Why we need sentence-level representations?
2. Pre-BERT methods
3. How to obtain sentence-level representations from BERTs?
 - a. Post-processing Methods
4. Contrastive Learning Methods:
 - a. Designed Positives
 - b. **Generating Positives**
 - c. Bootstrapping Methods
 - d. Dropout Augmentations
 - e. Equivariant Contrastive Learning
 - f. Prompting
 - g. Ranking-based Methods
5. Conclusion

Datasets from Instructions (DINO

- 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.”

Datasets from Instructions (DINO

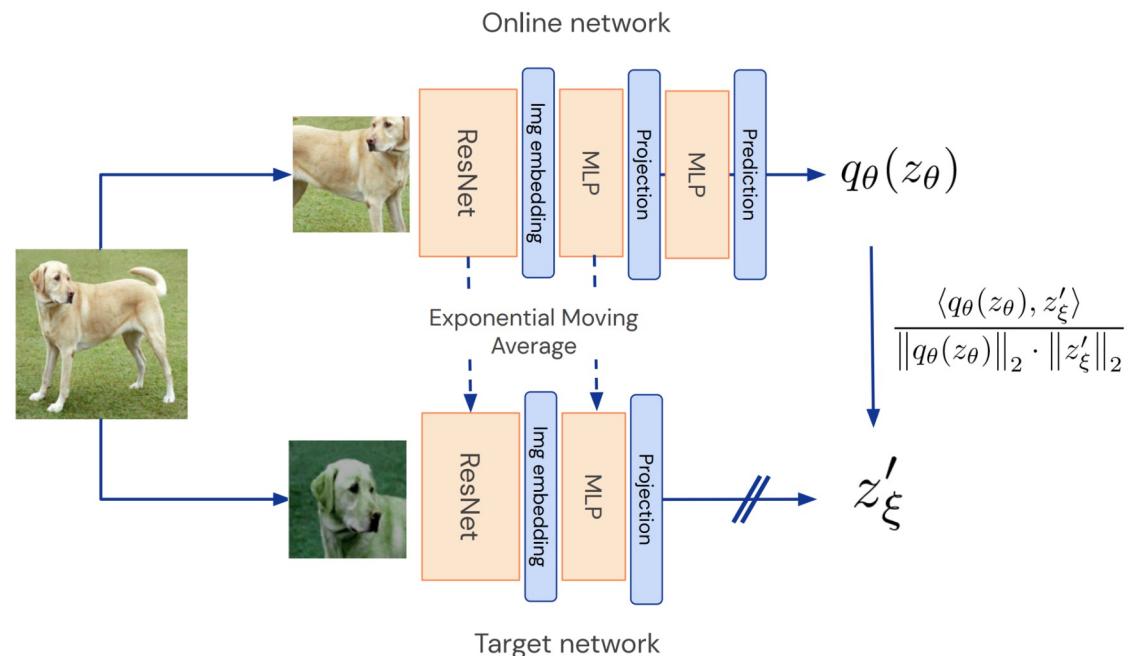
	Model	UD	STS12	STS13	STS14	STS15	STS16	STSb	SICK	Avg.	
sup.	InferSent, Glove	–	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01	
	USE	–	64.49	67.80	64.61	76.83	73.18	74.92	76.69	71.22	
	S-BERT (base)	–	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89	
	S-RoBERTa (base)	–	<u>71.54</u>	72.49	70.80	78.74	73.69	77.77	<u>74.46</u>	74.21	
unsup.	Avg. GloVe	–	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32	
	Avg. BERT	–	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81	
	BERT CLS	–	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19	
	Zhang et al. (2020)	NLI	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58	
	Li et al. (2020) BERT-flow	NLI	59.54	64.69	64.66	72.92	71.84	58.56	65.44	65.38	
	Li et al. (2020)	STS	63.48	72.14	68.42	73.77	75.37	70.72	63.11	69.57	
	DINO (STS-  -x ₁ x ₂)	–	64.87	78.30	66.38	79.60	76.47	76.51	74.26	73.77	
	DINO (STS-  -x ₂)	STS	70.27	81.26	71.25	80.49	77.18	77.82	68.09	75.20	
			DeCLUTR (BERT-base)	63.56	72.58	71.70	79.95	79.59	79.39	78.62	75.06
			ConSERT (BERT-base)	64.64	78.49	69.07	79.72	75.95	73.97	67.31	72.74

Outline of Part 3

1. Why we need sentence-level representations?
2. Pre-BERT methods
3. How to obtain sentence-level representations from BERTs?
 - a. Post-processing Methods
4. Contrastive Learning Methods:
 - a. Designed Positives
 - b. Generating Positives
 - c. **Bootstrapping Methods**
 - d. Dropout Augmentations
 - e. Equivariant Contrastive Learning
 - f. Prompting
 - g. Ranking-based Methods
5. Conclusion

BYOL

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



BYOL for sentence representations

- Back-Translation as positive pairs

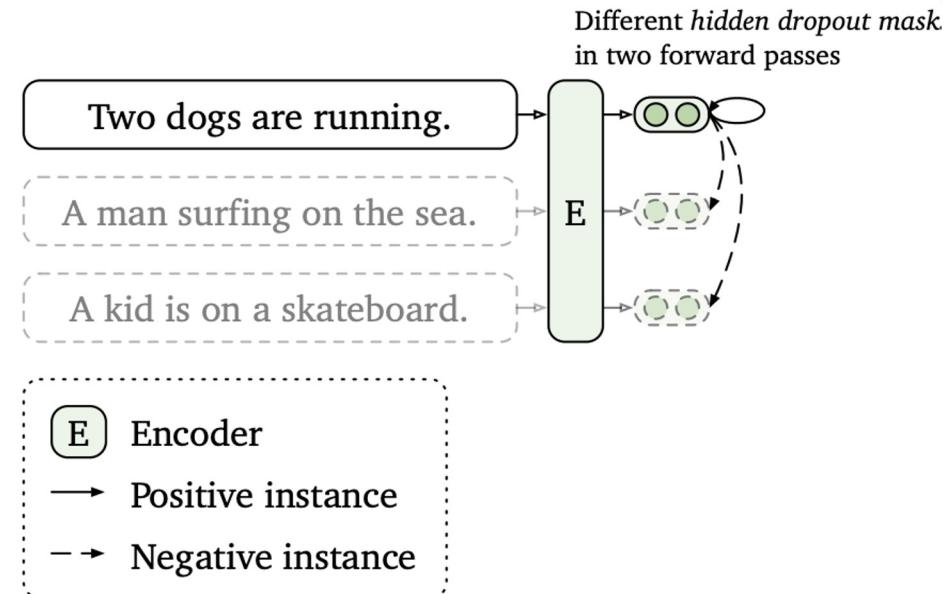
Model	STS-12	STS-13	STS-14	STS-15	STS-16	STS-B	SICK-R	Avg.
<i>Unsupervised methods</i>								
Unigram-TFIDF [†]	-	-	58.00	-	-	-	52.00	-
SDAE [†]	-	-	12.00	-	-	-	46.00	-
SkipThought [†]	-	-	27.00	-	-	-	57.00	-
FastSent [†]	-	-	63.00	-	-	-	61.00	-
GloVe avg. [‡]	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
BERT avg. [‡]	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT [CLS] [‡]	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
BERT-mlm	48.86	64.76	56.97	70.86	64.65	64.33	67.76	62.60
IS-BERT*	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
BERT-flow [○]	59.54	64.69	64.66	72.92	71.84	58.56	65.44	65.38
Ours: BSL-BERT	67.83	71.40	66.88	79.97	73.97	73.74	70.40	72.03
Ours: BSL-RoBERTa	68.47	72.41	68.48	78.50	72.77	78.77	69.97	72.76
DeCLUTR (BERT-base)	63.56	72.58	71.70	79.95	79.59	79.39	78.62	75.06
ConSERT (BERT-base)	64.64	78.49	69.07	79.72	75.95	73.97	67.31	72.74
DINO (RoBERTa-base)	70.27	81.26	71.25	80.49	77.18	77.82	68.09	75.20

Outline of Part 3

1. Why we need sentence-level representations?
2. Pre-BERT methods
3. How to obtain sentence-level representations from BERTs?
 - a. Post-processing Methods
4. Contrastive Learning Methods:
 - a. Designed Positives
 - b. Generating Positives
 - c. Bootstrapping Methods
 - d. **Dropout Augmentations**
 - e. Equivariant Contrastive Learning
 - f. Prompting
 - g. Ranking-based Methods
5. Conclusion

SimCSE (Unsupervised)

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



Results (STS)

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
<i>Unsupervised models</i>								
GloVe embeddings (avg.)♦	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
BERT _{base} (first-last avg.)	39.70	59.38	49.67	66.03	66.19	53.87	62.06	56.70
BERT _{base} -flow	58.40	67.10	60.85	75.16	71.22	68.66	64.47	66.55
BERT _{base} -whitening	57.83	66.90	60.90	75.08	71.31	68.24	63.73	66.28
IS-BERT _{base} ♡	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
CT-BERT _{base}	61.63	76.80	68.47	77.50	76.48	74.31	69.19	72.05
* SimCSE-BERT _{base}	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
RoBERTa _{base} (first-last avg.)	40.88	58.74	49.07	65.63	61.48	58.55	61.63	56.57
RoBERTa _{base} -whitening	46.99	63.24	57.23	71.36	68.99	61.36	62.91	61.73
DeCLUTR-RoBERTa _{base}	52.41	75.19	65.52	77.12	78.63	72.41	68.62	69.99
* SimCSE-RoBERTa _{base}	70.16	81.77	73.24	81.36	80.65	80.22	68.56	76.57
* SimCSE-RoBERTa _{large}	72.86	83.99	75.62	84.77	81.80	81.98	71.26	78.90
DeCLUTR (BERT-base)	63.56	72.58	71.70	79.95	79.59	79.39	78.62	75.06
ConSERT (BERT-base)	64.64	78.49	69.07	79.72	75.95	73.97	67.31	72.74
DINO (RoBERTa-base)	70.27	81.26	71.25	80.49	77.18	77.82	68.09	75.20

Why Dropout?

- Better than crop, word deletion and replacement
- Simple but super effective

Data augmentation	STS-B		
None (unsup. SimCSE)	82.5		
Crop	10%	20%	30%
	77.8	71.4	63.6
Word deletion	10% 20% 30%		
	75.9	72.2	68.2
Delete one word	75.9		
w/o dropout	74.2		
Synonym replacement	77.4		
MLM 15%	62.2		

Why Self-Prediction?

	STS-B results	share encoder	dual encoder
Training objective		f_θ	$(f_{\theta_1}, f_{\theta_2})$
QuickThoughts (pos: Next Sentence)	Next sentence	66.8	67.7
Self-Prediction (pos: Same Sentence)	Next 3 sentences	68.7	69.7
	Delete one word	74.8	70.4
	Unsupervised SimCSE	79.1	70.7

How to Dropout?

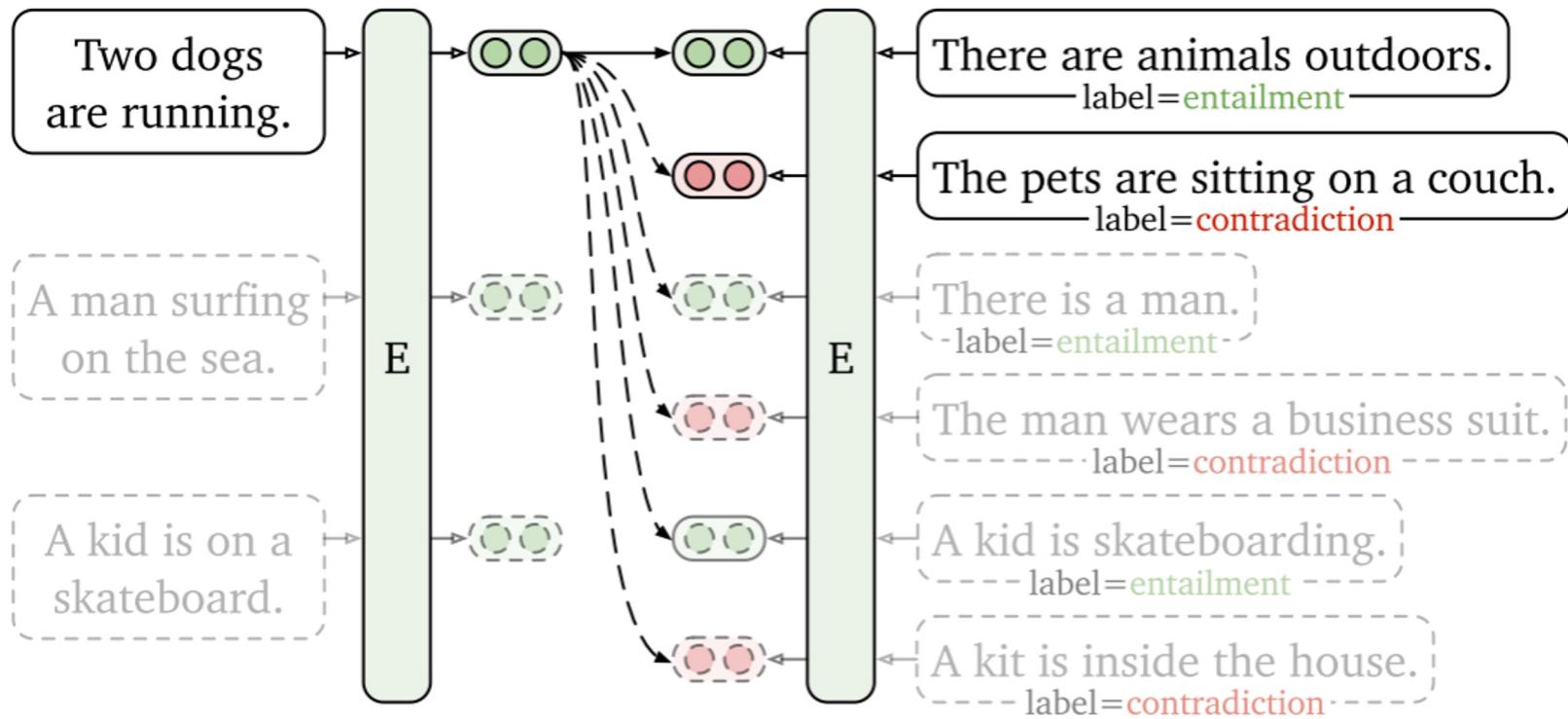
p	0.0	0.01	0.05	0.1
STS-B	64.9	69.5	78.0	79.1

p	0.15	0.2	0.5	<i>Fixed 0.1</i>
STS-B	78.6	78.2	67.4	45.2

- **Fixed 0.1:** apply the same dropout mask for two inputs
-> leading to mode collapsing
- Best: Two different dropout masks with $p = 0.1$

Supervised SimCSE

(b) Supervised SimCSE



Supervised SimCSE

back-translation paraphrase

use contradict as negative examples

Dataset	sample	full
Unsup. SimCSE (1m)	-	82.5
QQP (134k)	81.8	81.8
Flickr30k (318k)	81.5	81.4
ParaNMT (5m)	79.7	78.7
SNLI+MNLI		
entailment (314k)	84.1	84.9
neutral (314k) ⁸	82.6	82.9
contradiction (314k)	77.5	77.6
all (942k)	81.7	81.9
SNLI+MNLI		
entailment + hard neg.	-	86.2
+ ANLI (52k)	-	85.0

Supervised SimCSE

Unsupervised SImCSE v.s Supervised SimCSE

Supervised model is still performs much better

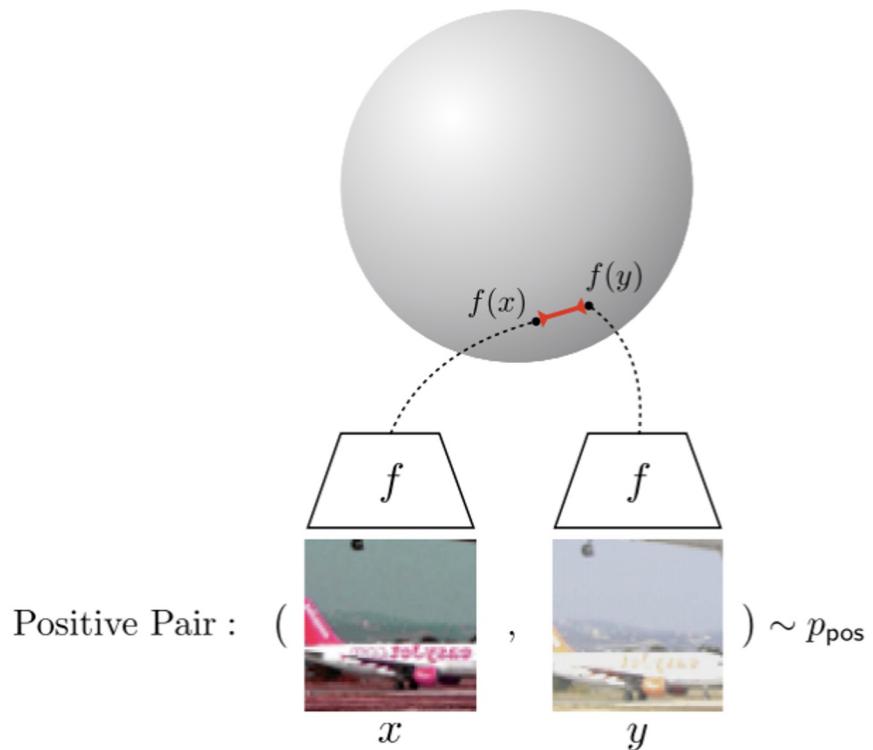
-> Large space for unsupervised models to improve

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
<i>Unsupervised models</i>								
* SimCSE-BERT _{base}	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
<i>Supervised models</i>								
* SimCSE-BERT _{base}	75.30	84.67	80.19	85.40	80.82	84.25	80.39	81.57

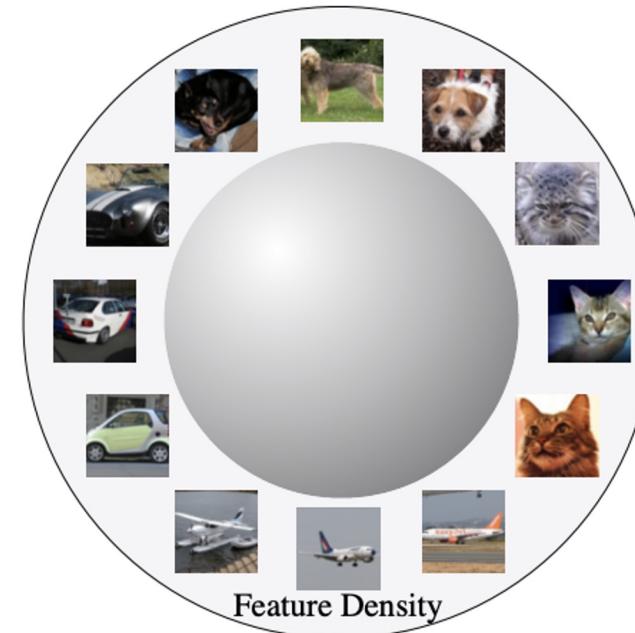
Alignment & Uniformity

$$\ell_{\text{align}} \triangleq \mathbb{E}_{(x, x^+) \sim p_{\text{pos}}} \|f(x) - f(x^+)\|^2.$$

$$\ell_{\text{uniform}} \triangleq \log \mathbb{E}_{\substack{x, y \sim i.i.d. \\ p_{\text{data}}}} e^{-2\|f(x) - f(y)\|^2},$$



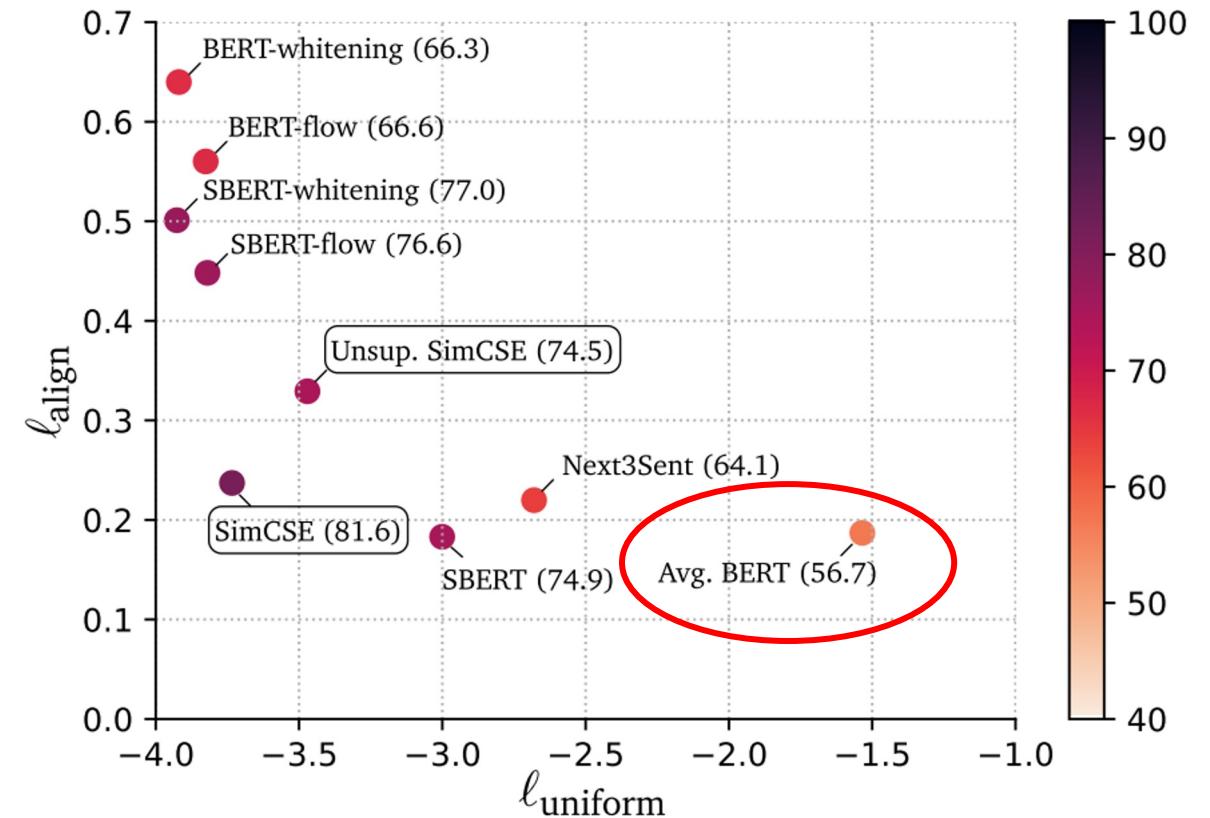
Alignment: Similar samples have similar features.



Uniformity: Preserve maximal information.

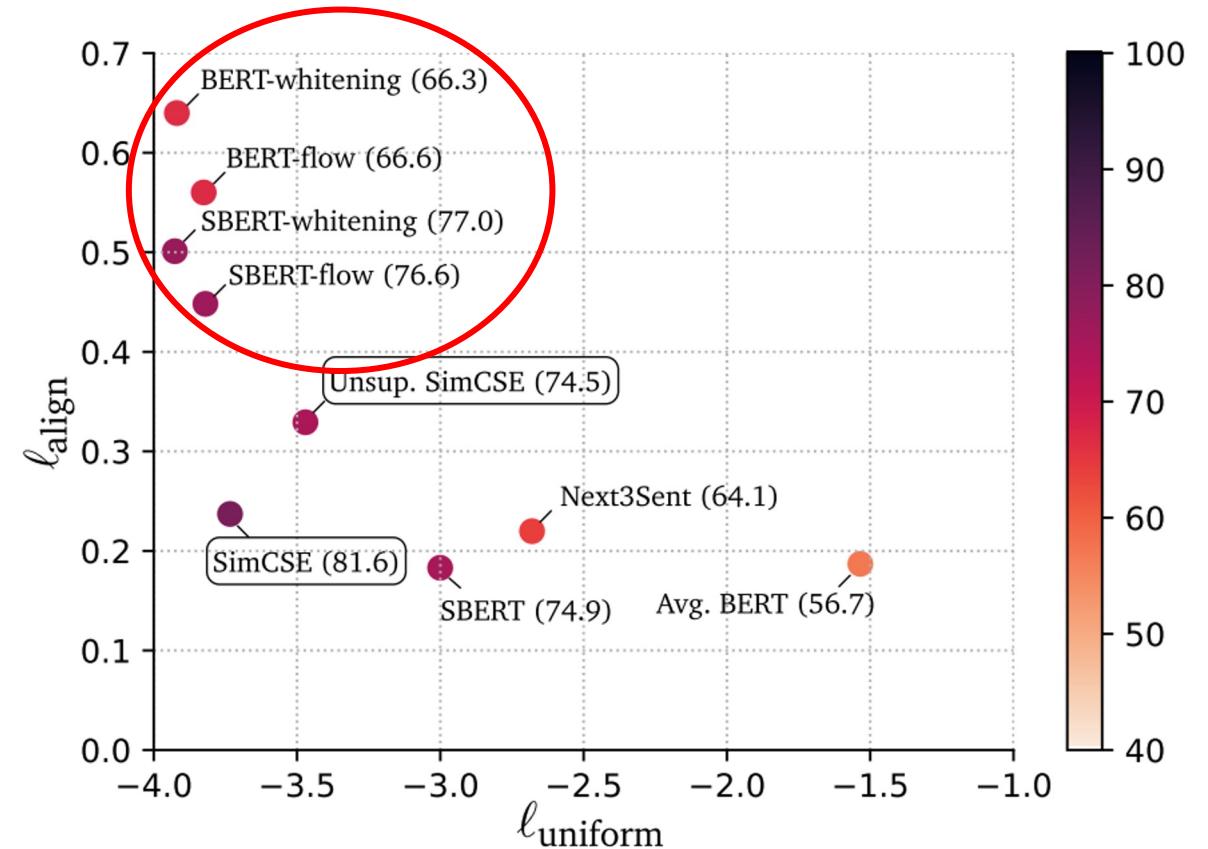
Analysis

- Pre-trained embedding:
good alignment
poor uniformity
=> anisotropic



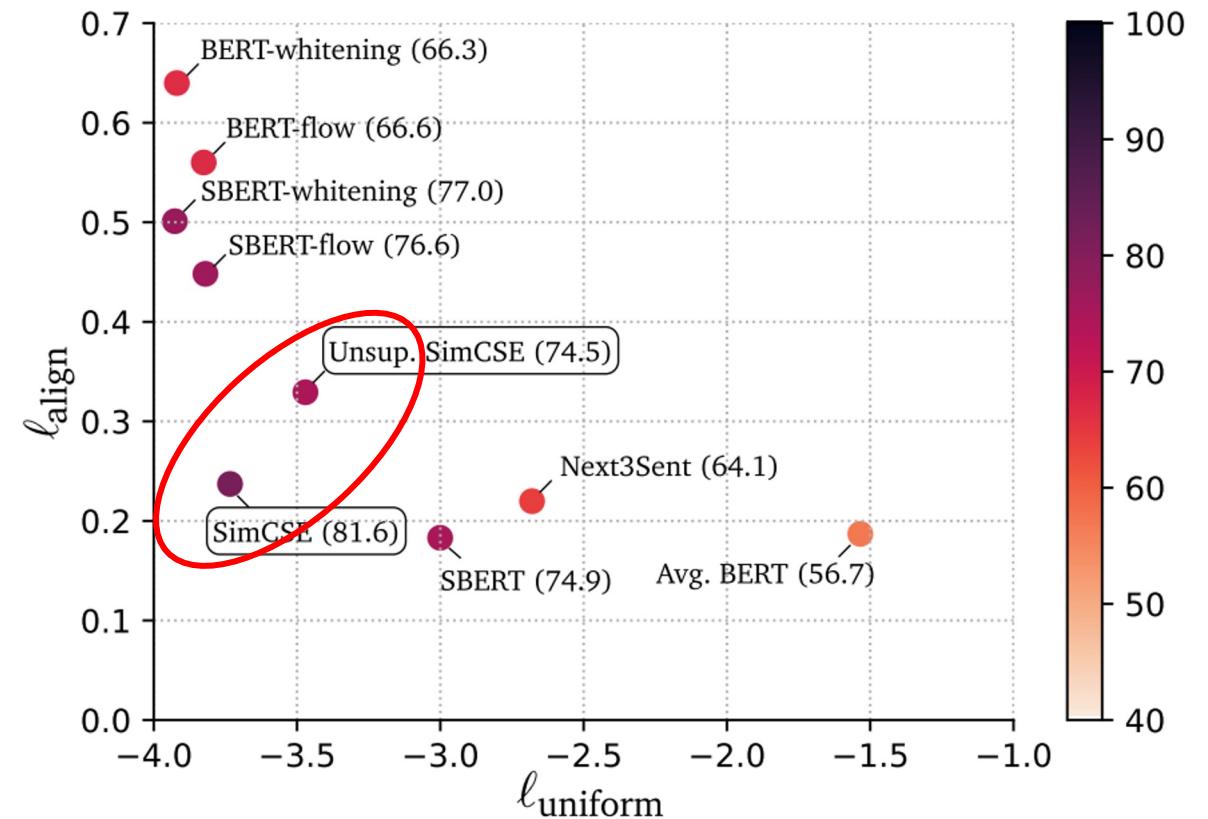
Analysis

- Pre-trained embedding:
good alignment
poor uniformity
=> anisotropic
- Post-processing methods
(BERT-whitening/flow):
good uniformity
poor alignment



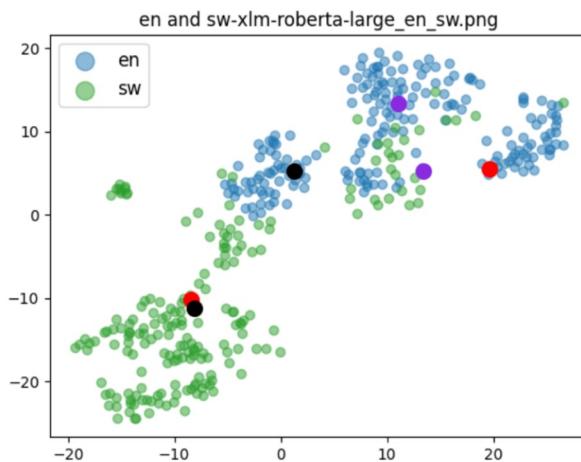
Analysis

- Pre-trained embedding:
good alignment
poor uniformity
=> anisotropic
- Post-processing methods
(BERT-whitening/flow):
good uniformity
poor alignment
- **SimCSE:**
Best of the both worlds

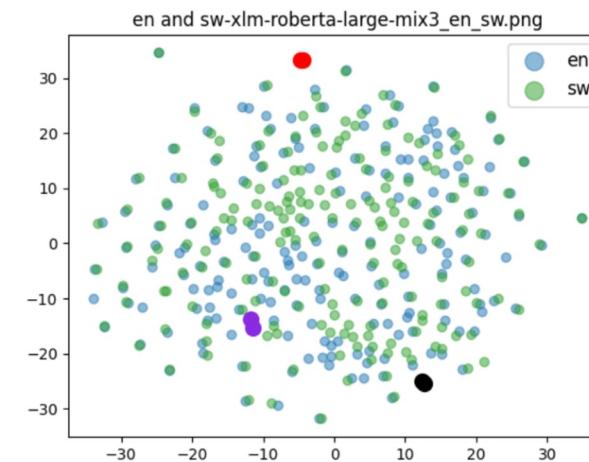


mSimCSE

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



(a) XLM-R without finetuning.



(b) XLM-R fintuned on English NLI data.

mSimCSE

- mSimCSE performs close to **supervised** multilingual sentence encoder such as LaBSE.

Models	BUCC	Tatoeba-14	Tatoeba-36
Unsupervised			
XLM-R	66.0	57.6	53.4
INFOXML	-	77.8	67.3
DuEAM	77.2	-	-
XLM-E	-	72.3	62.3
HiCTL	68.4	-	59.7
<i>mSimCSE_{en}</i>	87.5	82.0	78.0
English NLI supervised			
(Phang et al., 2020)	71.9	-	81.2
<i>mSimCSE_{en}</i>	93.6	89.9	87.7
Cross-lingual NLI supervised			
<i>mSimCSE_{en,fr}</i>	94.2	90.8	88.8
<i>mSimCSE_{en,fr,sw}</i>	94.3	93.3	90.3
<i>mSimCSE_{all}</i>	95.2	93.2	91.4
DuEAM	81.7	-	-
Fully Supervised			
LASER	92.9	95.3	84.4
LaBSE	93.5	95.3	95.0
<i>mSimCSE_{sw}</i>	86.8	87.7	86.3
<i>mSimCSE_{fr}</i>	87.1	87.9	85.9
<i>mSimCSE_{sw,fr}</i>	88.8	90.2	88.3
<i>mSimCSE_{sw,fr}+NLI</i>	93.6	91.9	90.0

Outline of Part 3

1. Why we need sentence-level representations?
2. Pre-BERT methods
3. How to obtain sentence-level representations from BERTs?
 - a. Post-processing Methods
4. Contrastive Learning Methods:
 - a. Designed Positives
 - b. Generating Positives
 - c. Bootstrapping Methods
 - d. Dropout Augmentations
 - e. **Equivariant Contrastive Learning**
 - f. Prompting
 - g. Ranking-based Methods
5. Conclusion

What we've learned from SimCSE?

Augmentations for natural language is hard:

- Word deletion is not a good augmentation
- Synonym replacement is not a good augmentation
- Sentence cropping is not a good augmentation
- Back-translation is not a good augmentation
- ...

They are all outperformed by simply changing dropout masks :(

Let's take a step back...

Q: Why do we need positive pairs in contrastive learning?

A: to make the representations invariant to these kinds of augmentations.

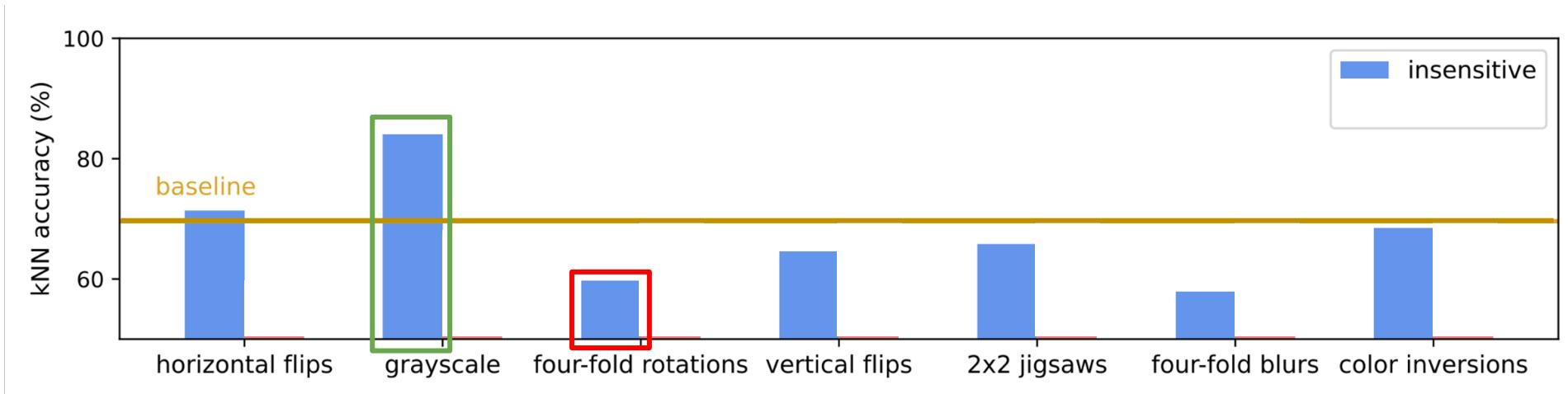
Problem:

It's hard to produce semantically similar augmentations for natural language.
Making the representations invariant to augmentations will hurt performance.

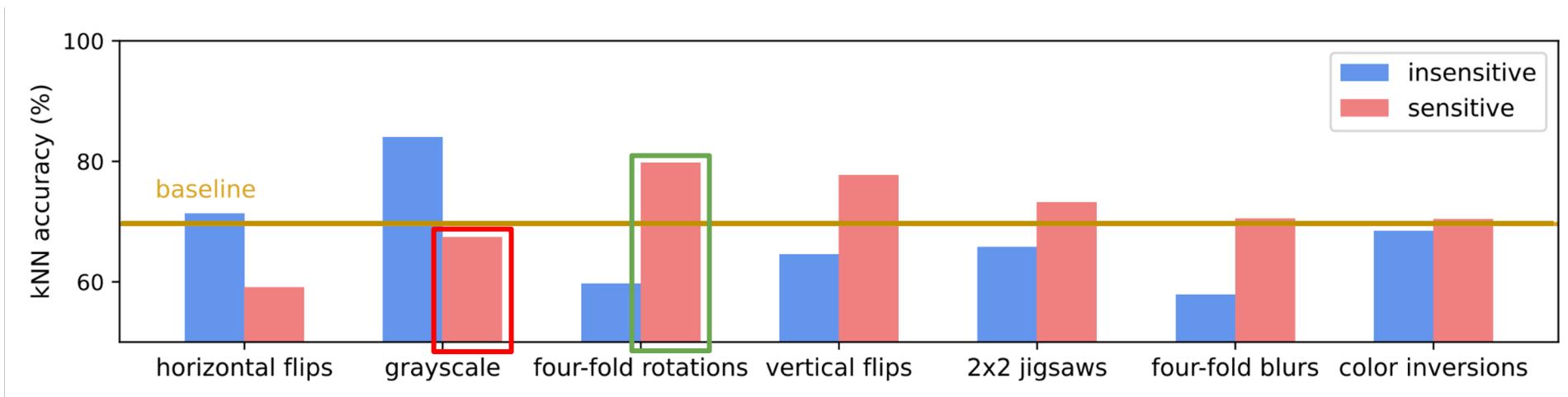
Q: Is there another way to utilize augmentations...?

A: we can make the representations be aware of,
but not necessarily invariant to the augmentations.

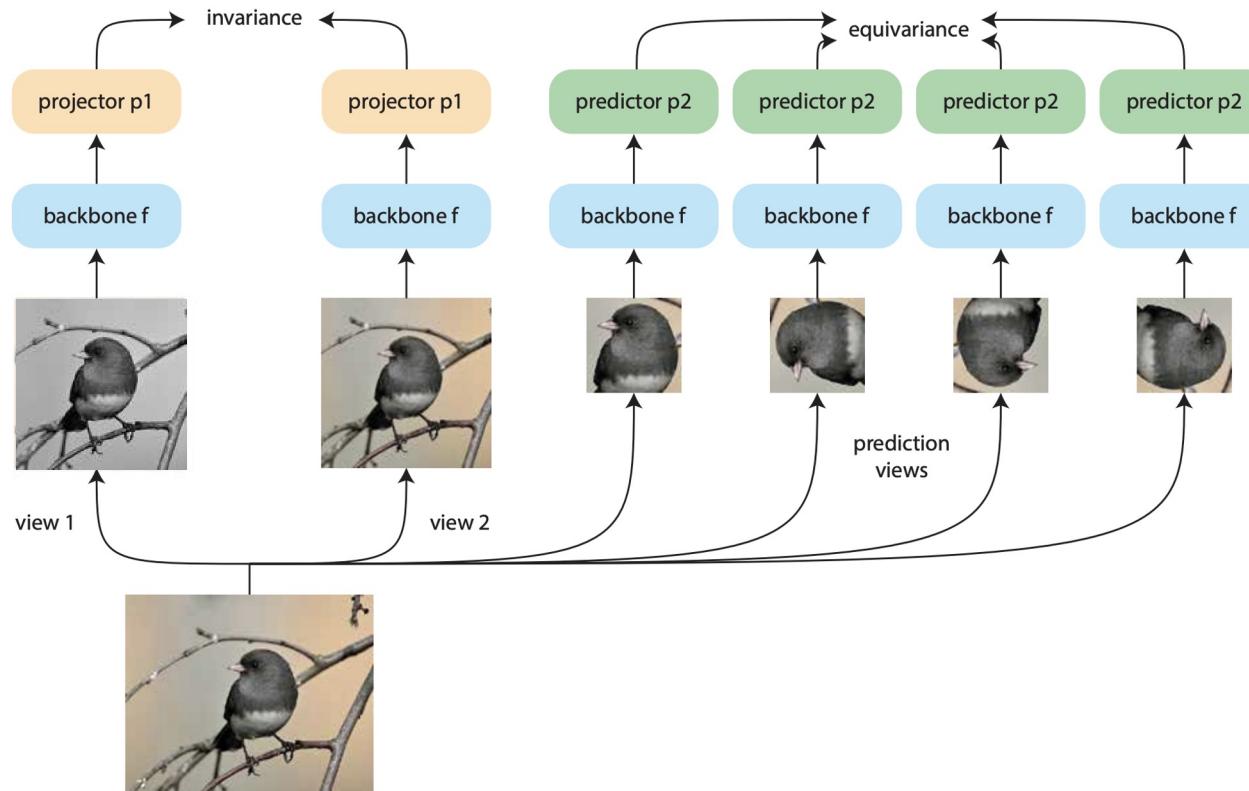
Background: Equivariant Contrastive Learning



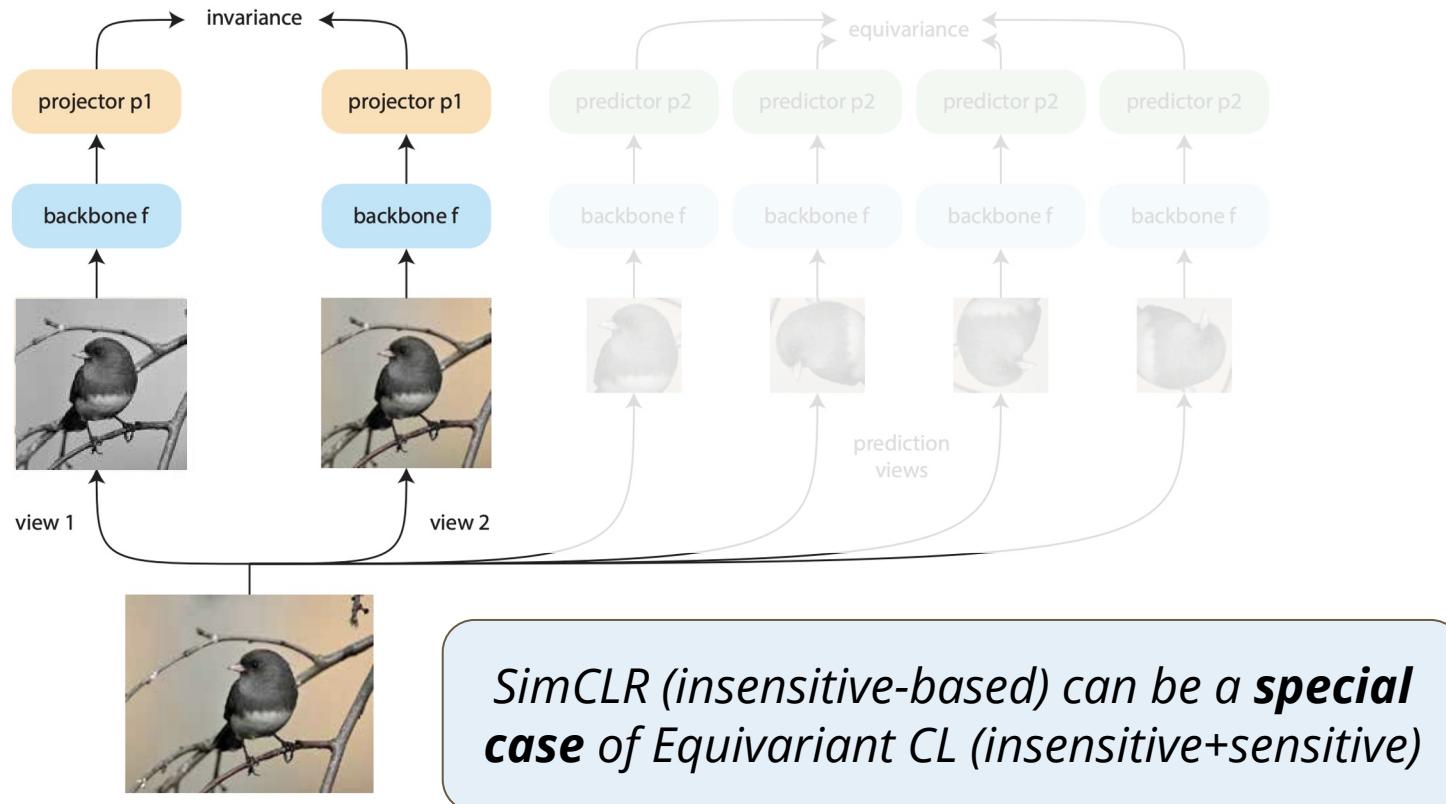
Background: Equivariant Contrastive Learning



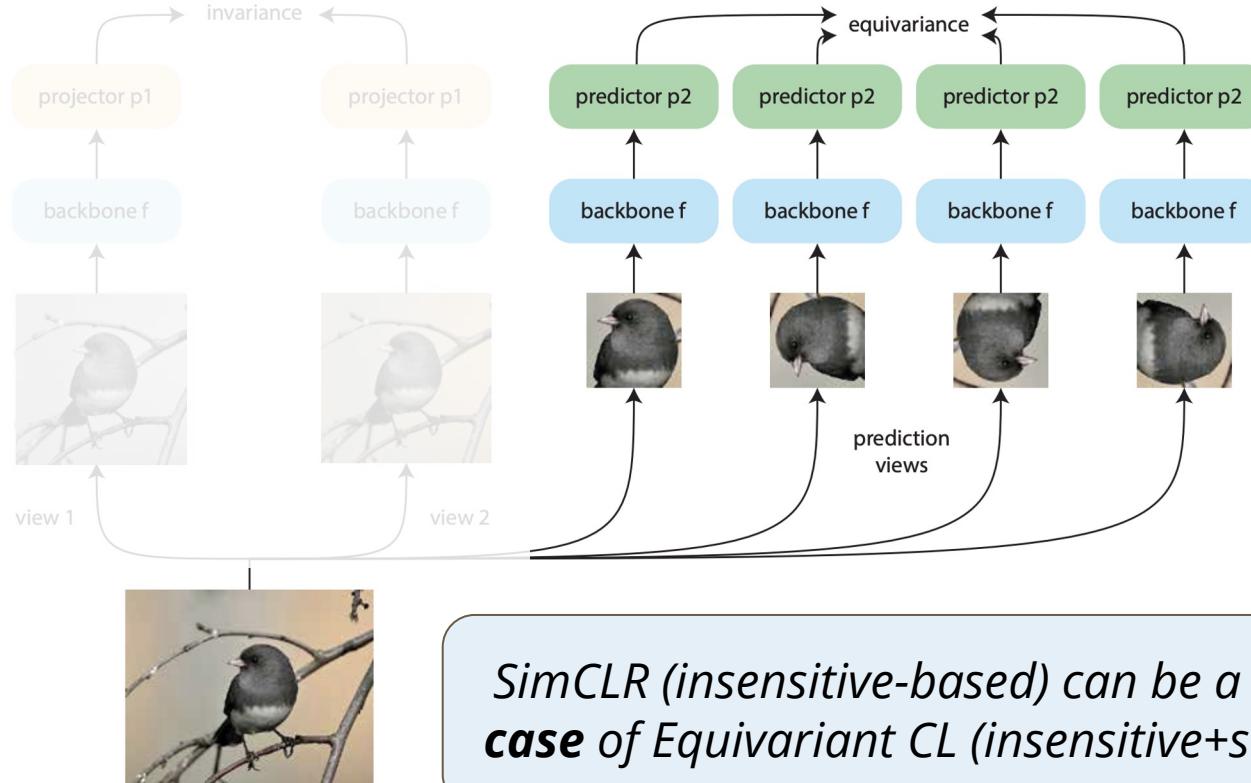
Background: Equivariant Contrastive Learning



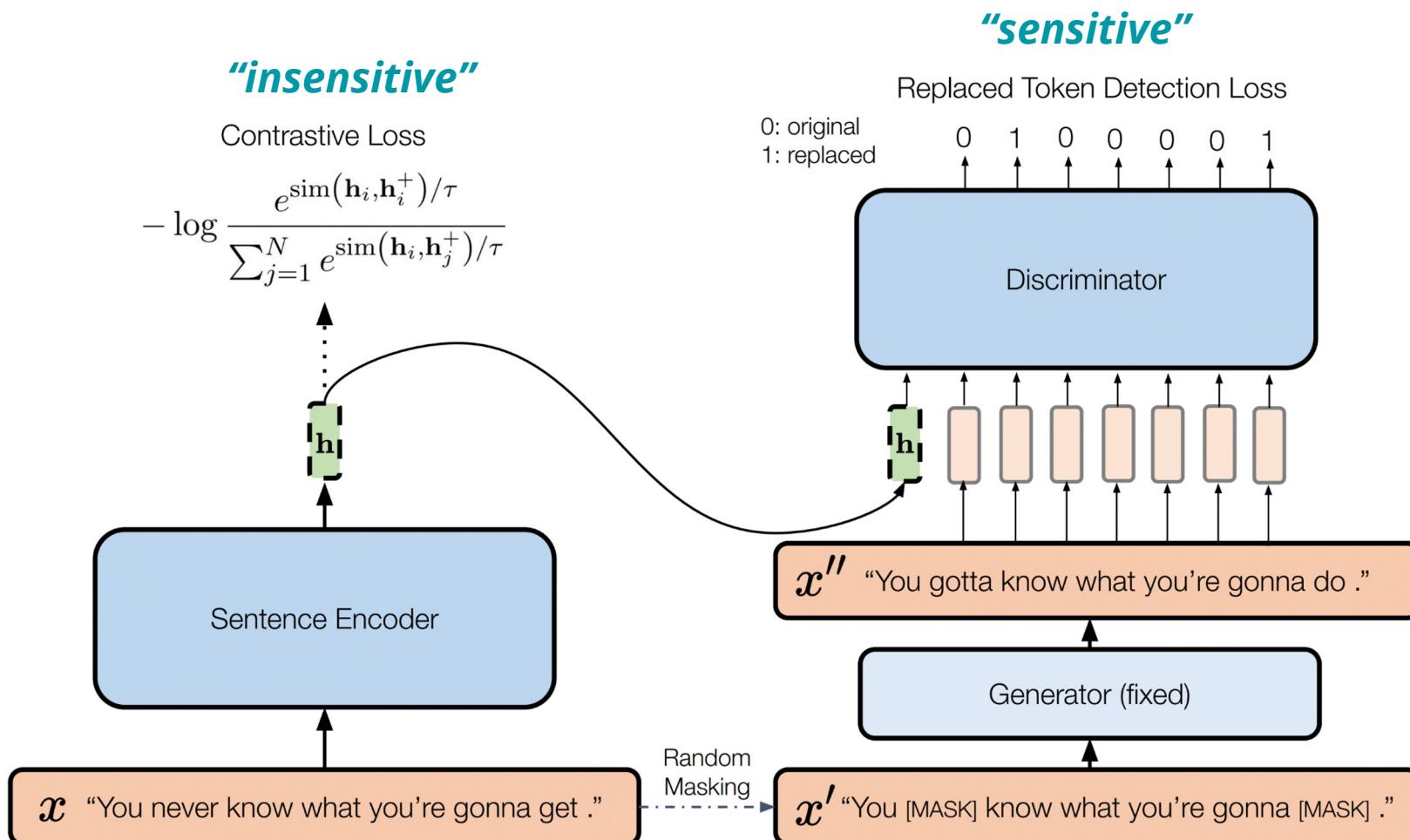
Background: Equivariant Contrastive Learning



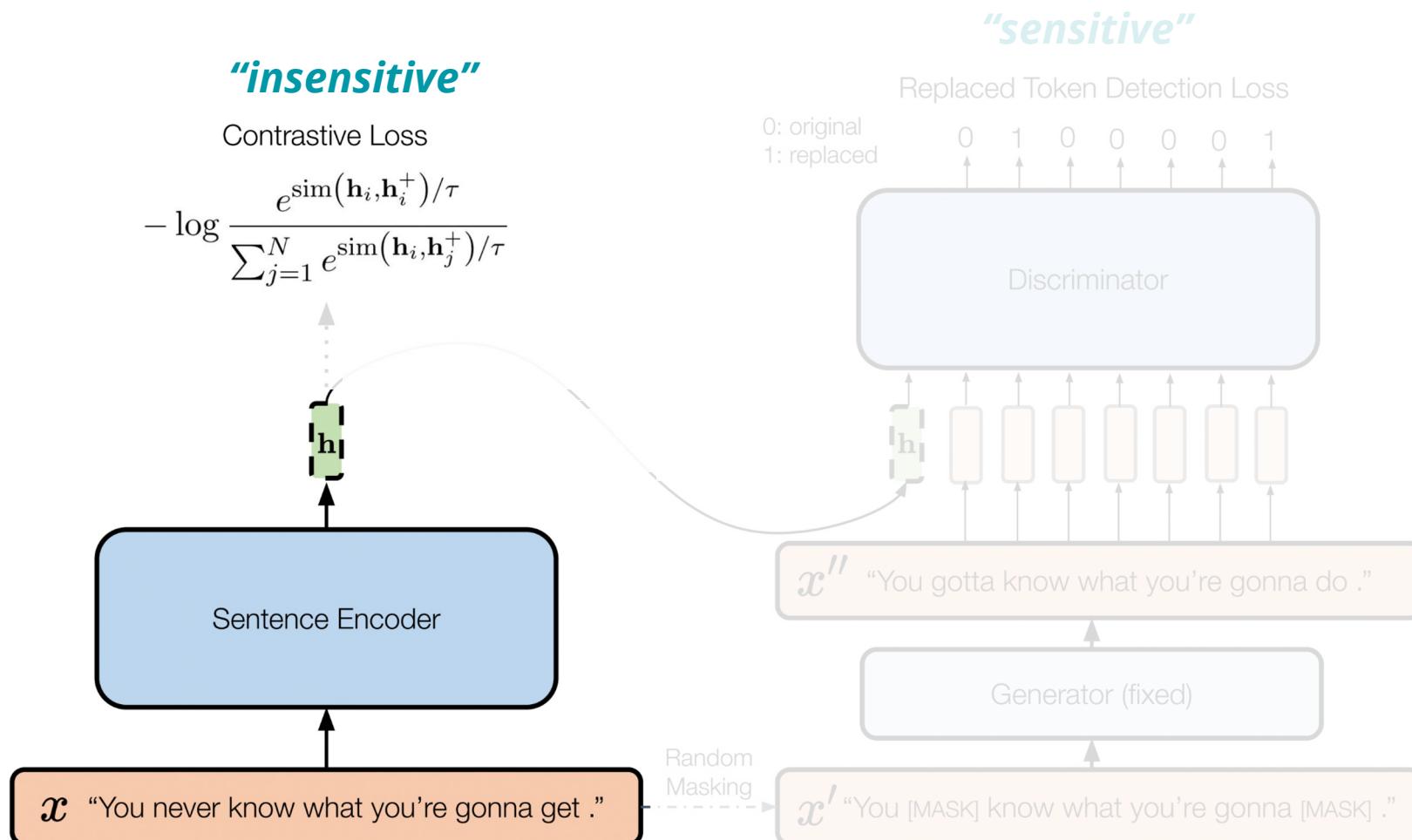
Background: Equivariant Contrastive Learning



DiffCSE

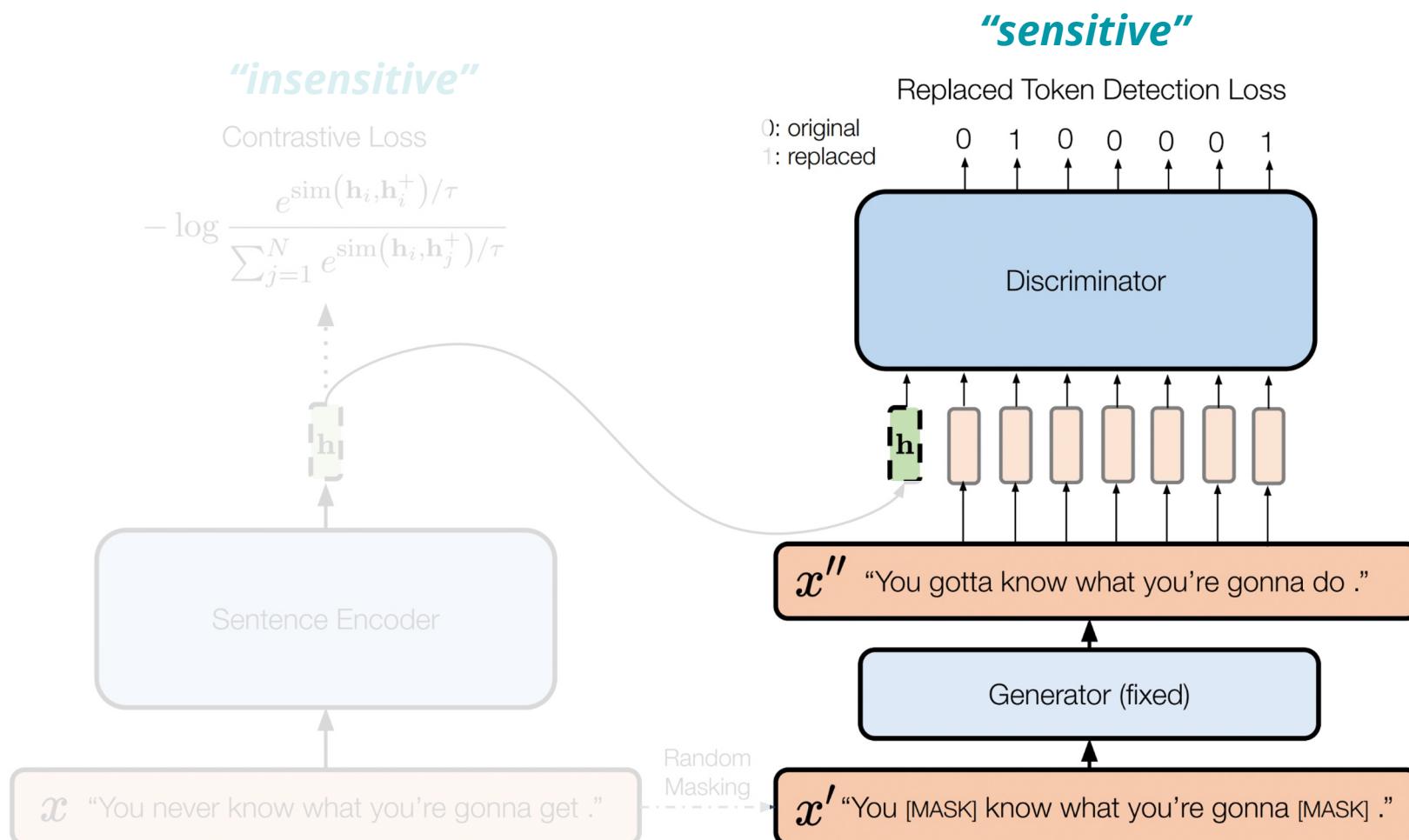


DiffCSE

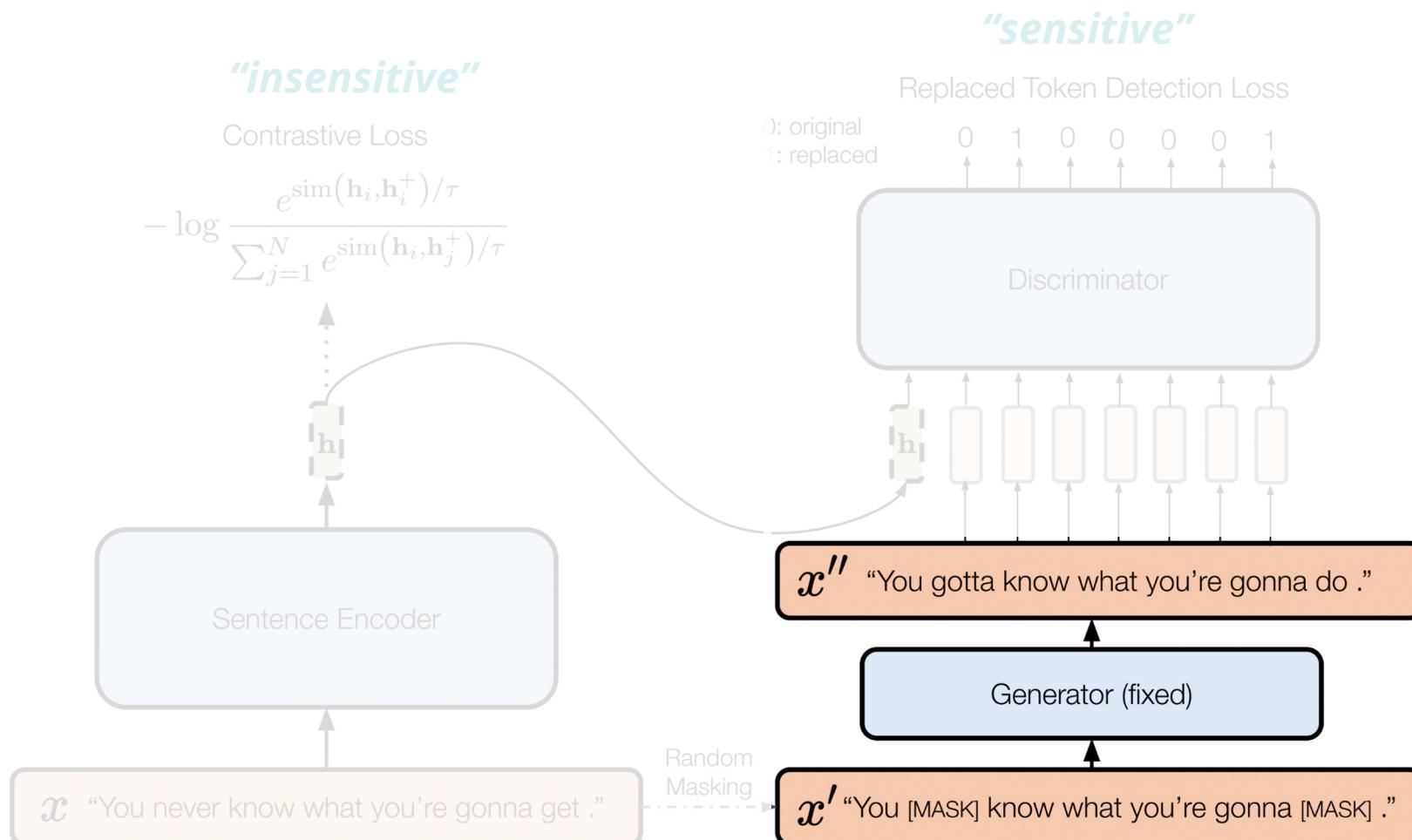


Chuang, Yung-Sung, et al. "DiffCSE: Difference-based Contrastive Learning for Sentence Embeddings. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2022.

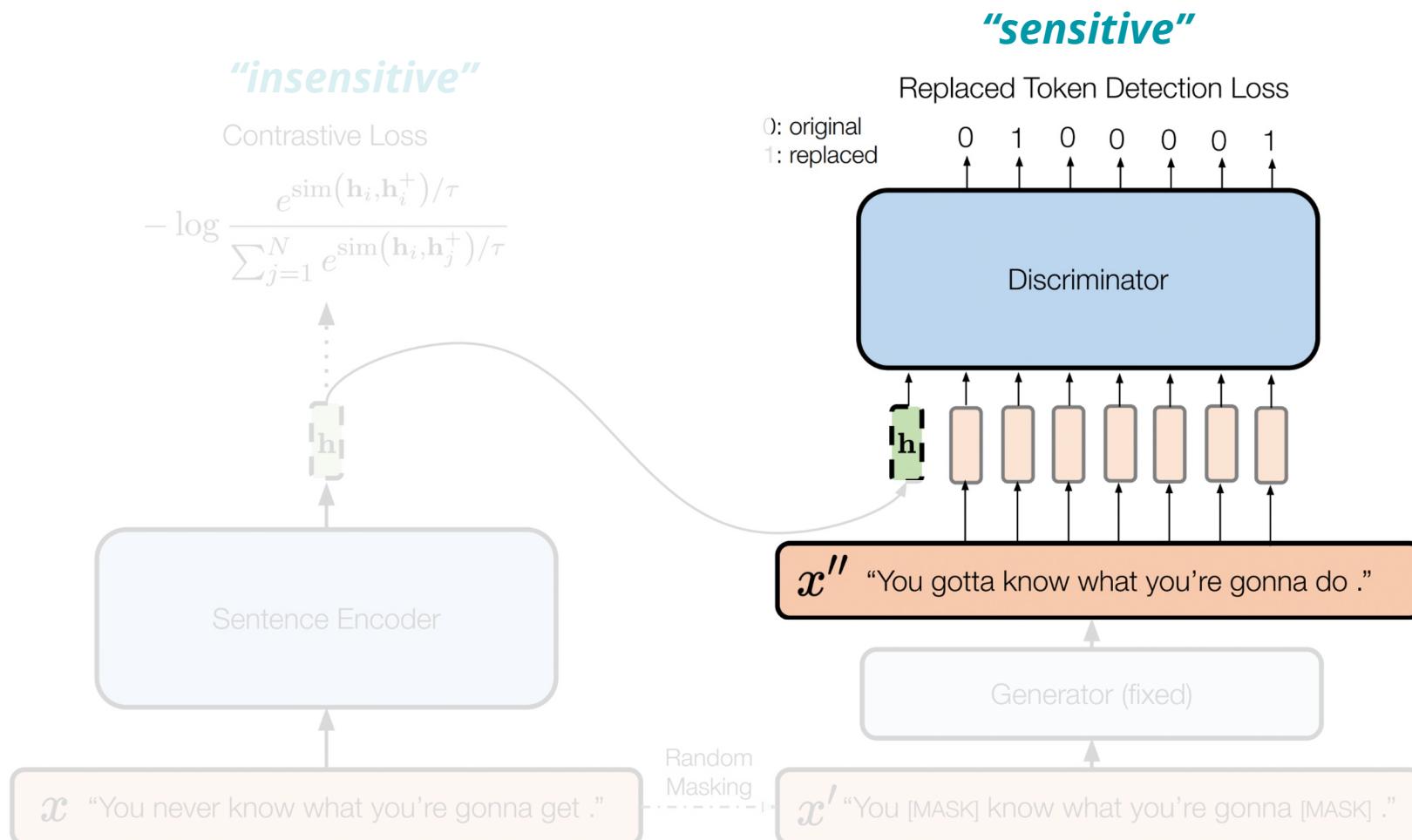
DiffCSE



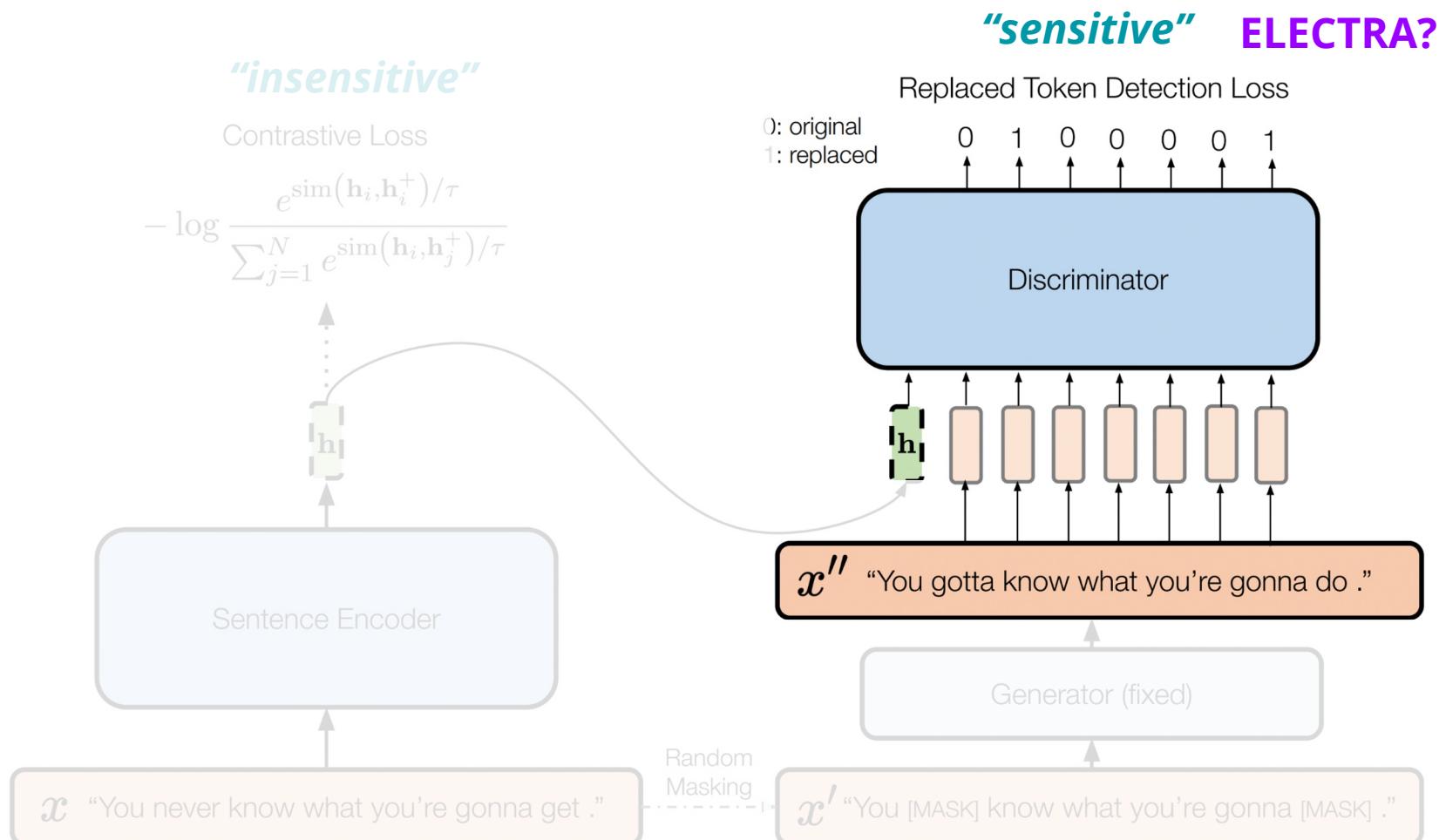
DiffCSE



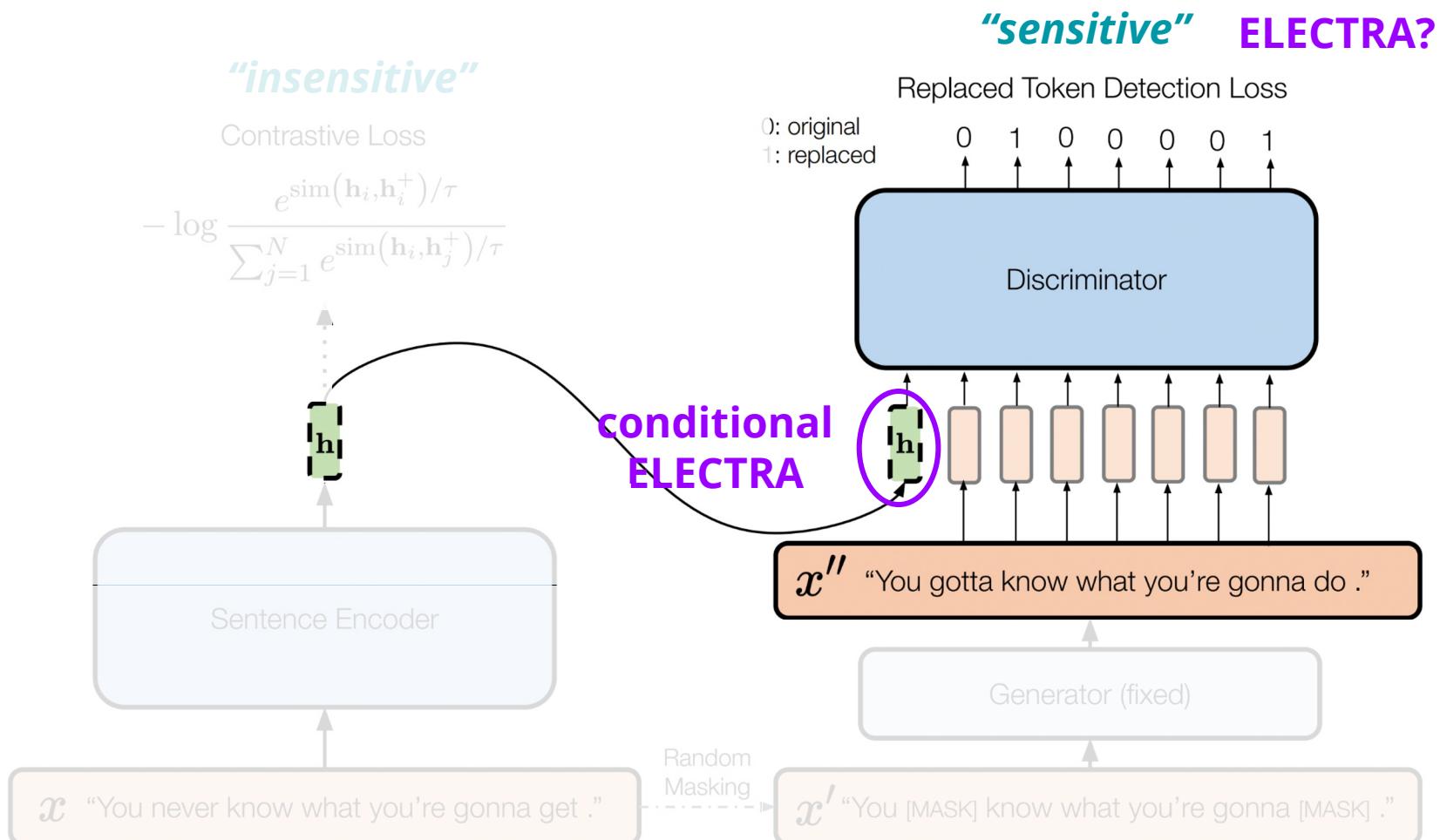
DiffCSE



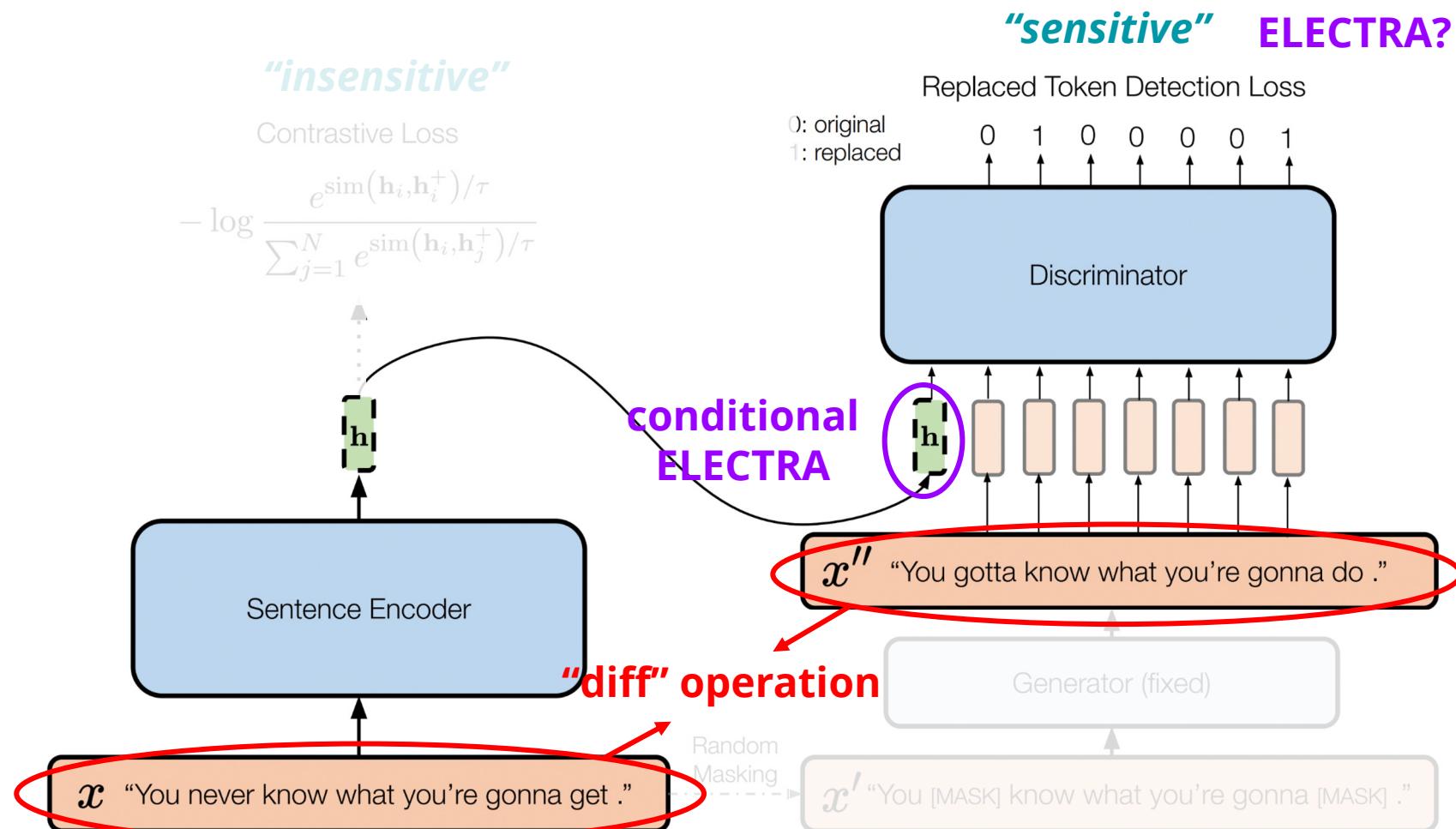
DiffCSE



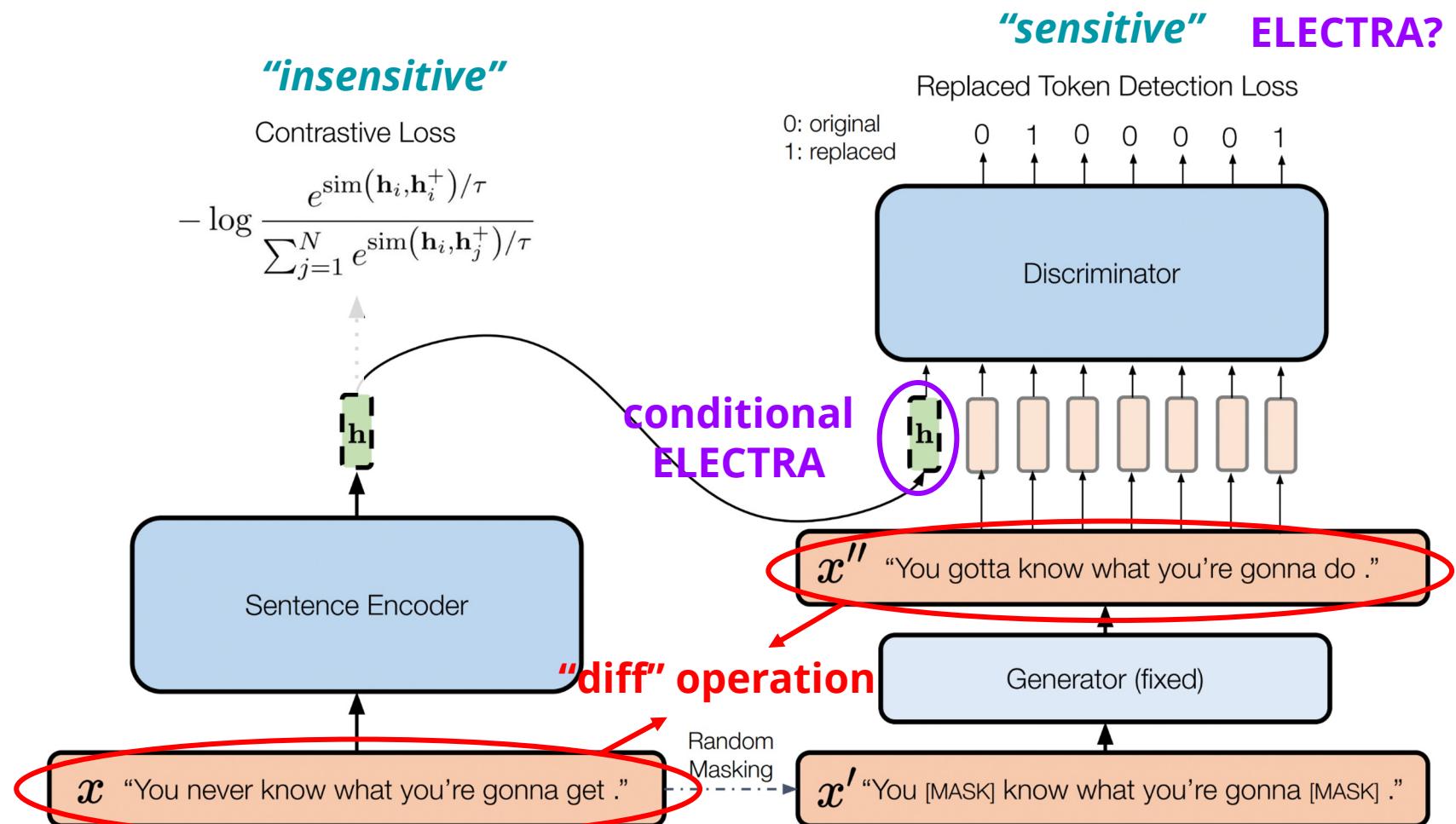
DiffCSE



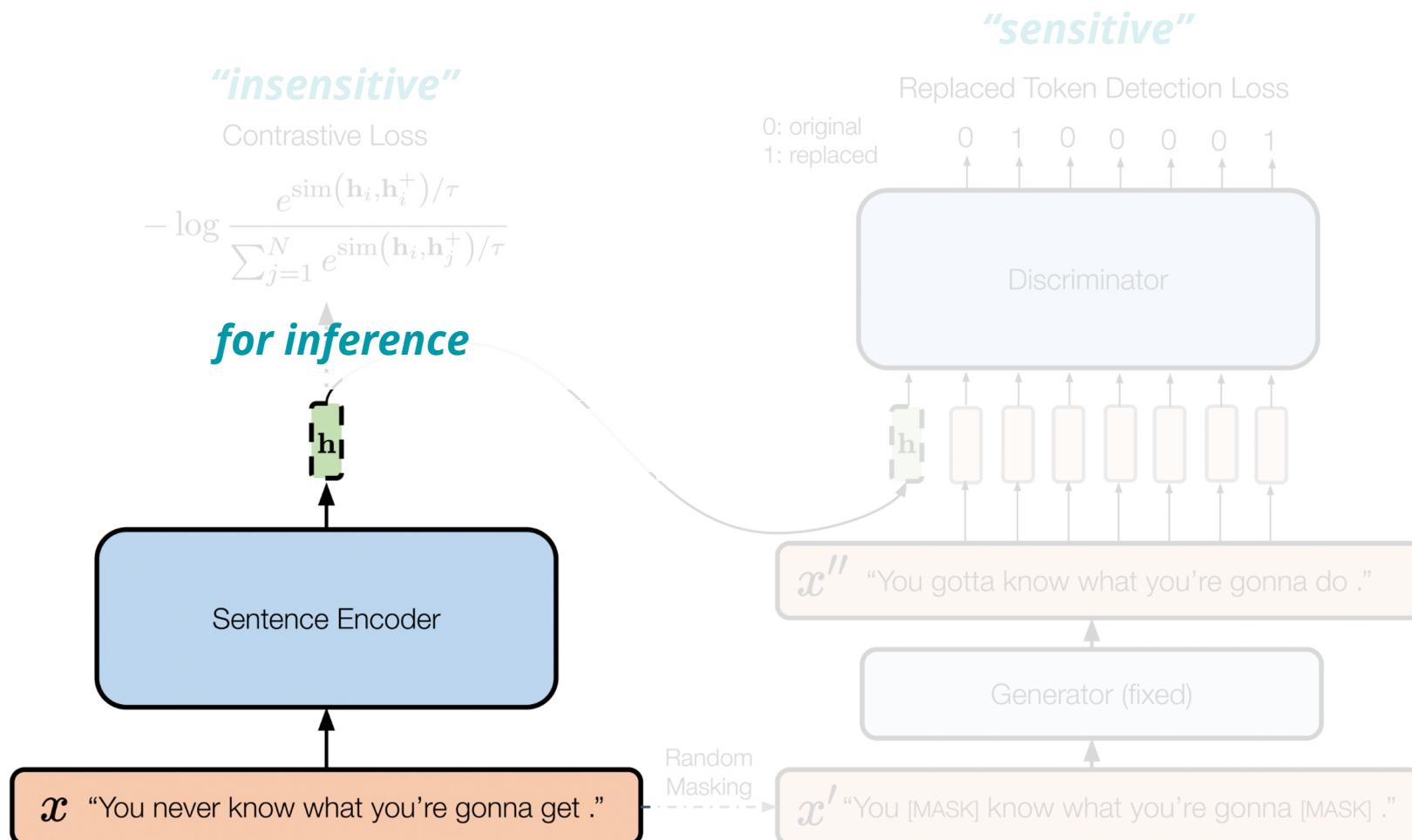
DiffCSE



DiffCSE



DiffCSE



Results (STS)

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
GloVe embeddings (avg.) ♣	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
BERT _{base} (first-last avg.) ◇	39.70	59.38	49.67	66.03	66.19	53.87	62.06	56.70
BERT _{base} -flow ◇	58.40	67.10	60.85	75.16	71.22	68.66	64.47	66.55
BERT _{base} -whitening ◇	57.83	66.90	60.90	75.08	71.31	68.24	63.73	66.28
IS-BERT _{base} ♦	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
CMLM-BERT _{base} ♪ (1TB data)	58.20	61.07	61.67	73.32	74.88	76.60	64.80	67.22
CT-BERT _{base} ◇	61.63	76.80	68.47	77.50	76.48	74.31	69.19	72.05
SG-OPT-BERT _{base} †	66.84	80.13	71.23	81.56	77.17	77.23	68.16	74.62
SimCSE-BERT _{base} ◇	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
* SimCSE-BERT _{base} (reproduce)	70.82	82.24	73.25	81.38	77.06	77.24	71.16	76.16
* DiffCSE-BERT _{base}	72.28	84.43	76.47	83.90	80.54	80.59	71.23	78.49
RoBERTa _{base} (first-last avg.) ◇	40.88	58.74	49.07	65.63	61.48	58.55	61.63	56.57
RoBERTa _{base} -whitening ◇	46.99	63.24	57.23	71.36	68.99	61.36	62.91	61.73
DeCLUTR-RoBERTa _{base} ◇	52.41	75.19	65.52	77.12	78.63	72.41	68.62	69.99
SimCSE-RoBERTa _{base} ◇	70.16	81.77	73.24	81.36	80.65	80.22	68.56	76.57
* SimCSE-RoBERTa _{base} (reproduce)	68.60	81.36	73.16	81.61	80.76	80.58	68.83	76.41
* DiffCSE-RoBERTa _{base}	70.05	83.43	75.49	82.81	82.12	82.38	71.19	78.21

Retrieval Results

SimCSE-BERT_{base}

Query: This is not a problem.

- 1) This is a big problem.
 - 2) You have a problem.
 - 3) I don't see why that should be a problem.
-

DiffCSE-BERT_{base}

- 1) I don't see why this could be a problem.
 - 2) I don't see why that should be a problem.
 - 3) This is a big problem.
-

Outline of Part 3

1. Why we need sentence-level representations?
2. Pre-BERT methods
3. How to obtain sentence-level representations from BERTs?
 - a. Post-processing Methods
4. Contrastive Learning Methods:
 - a. Designed Positives
 - b. Generating Positives
 - c. Bootstrapping Methods
 - d. Dropout Augmentations
 - e. Equivariant Contrastive Learning
 - f. **Prompting**
 - g. Ranking-based Methods
5. Conclusion

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

Template	STS-B dev.
<i>Searching for relationship tokens</i>	
[X] [MASK] .	39.34
[X] is [MASK] .	47.26
[X] mean [MASK] .	53.94
[X] means [MASK] .	63.56
<i>Searching for prefix tokens</i>	
This [X] means [MASK] .	64.19
This sentence of [X] means [MASK] .	68.97
This sentence of “[X]” means [MASK] .	70.19
This sentence : “[X]” means [MASK] .	73.44

PromptBERT

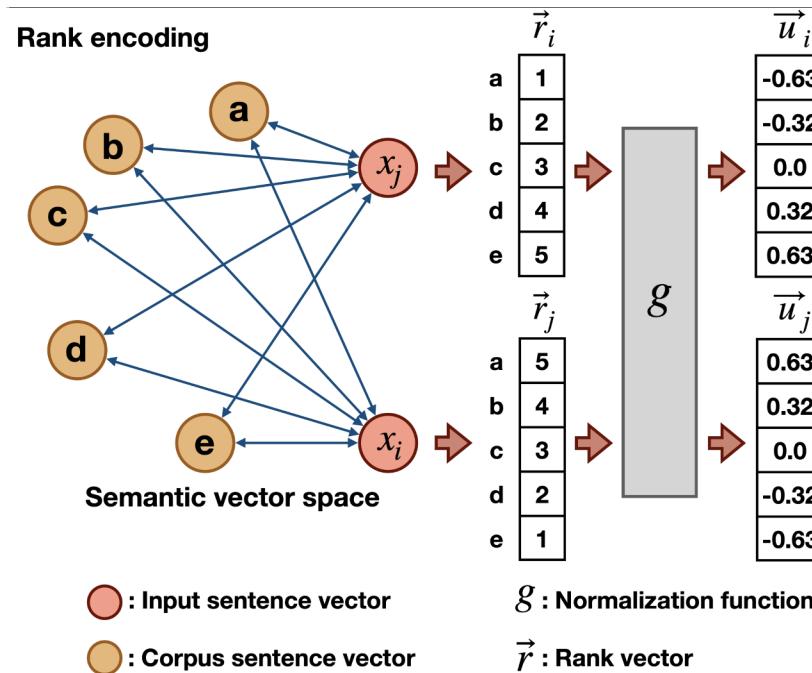
Method	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
<i>Unsupervised models</i>								
IS-BERT _{base} ¶	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
ConSERT _{base} ‡	64.64	78.49	69.07	79.72	75.95	73.97	67.31	72.74
SimCSE-BERT _{base} ‡	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
PromptBERT _{base}	71.56 _{±0.18}	84.58 _{±0.22}	76.98 _{±0.26}	84.47 _{±0.24}	80.60 _{±0.21}	81.60 _{±0.22}	69.87 _{±0.40}	78.54 _{±0.15}
RoBERTa _{base} -whitening†	46.99	63.24	57.23	71.36	68.99	61.36	62.91	61.73
SimCSE-RoBERTa _{base} †	70.16	81.77	73.24	81.36	80.65	80.22	68.56	76.57
PromptRoBERTa _{base}	73.94 _{±0.90}	84.74 _{±0.36}	77.28 _{±0.41}	84.99 _{±0.25}	81.74 _{±0.29}	81.88 _{±0.37}	69.50 _{±0.57}	79.15 _{±0.25}
DiffCSE (BERT-base)	72.28	84.43	76.47	83.90	80.54	80.59	71.23	78.49

Outline of Part 3

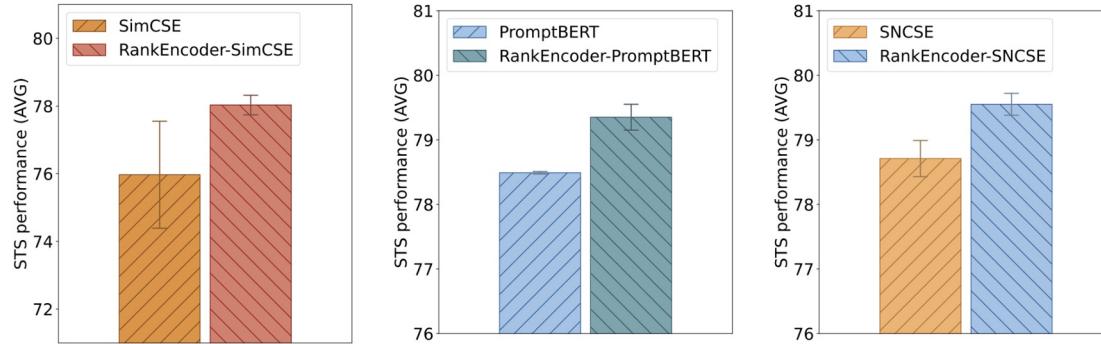
1. Why we need sentence-level representations?
2. Pre-BERT methods
3. How to obtain sentence-level representations from BERTs?
 - a. Post-processing Methods
4. Contrastive Learning Methods:
 - a. Designed Positives
 - b. Generating Positives
 - c. Bootstrapping Methods
 - d. Dropout Augmentations
 - e. Equivariant Contrastive Learning
 - f. Prompting
 - g. **Ranking-based Methods**
5. Conclusion

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



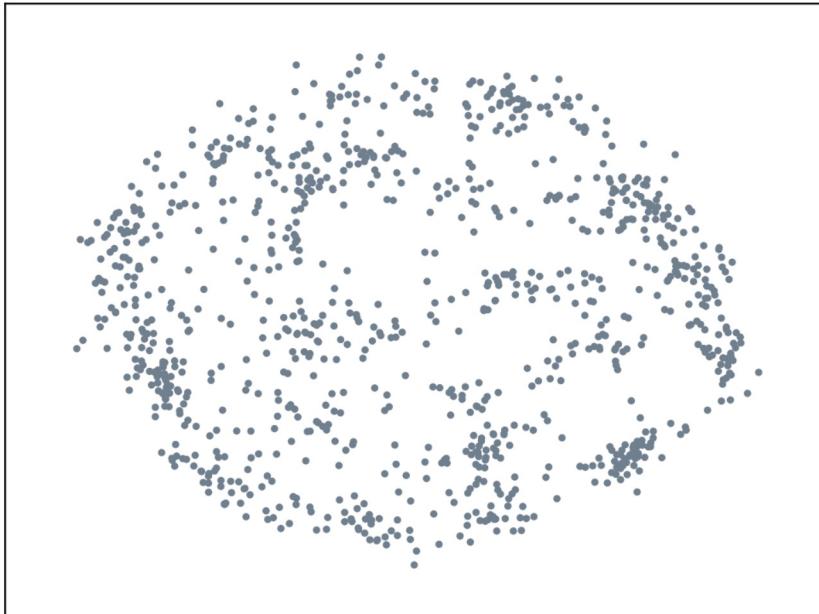
RankEncoder



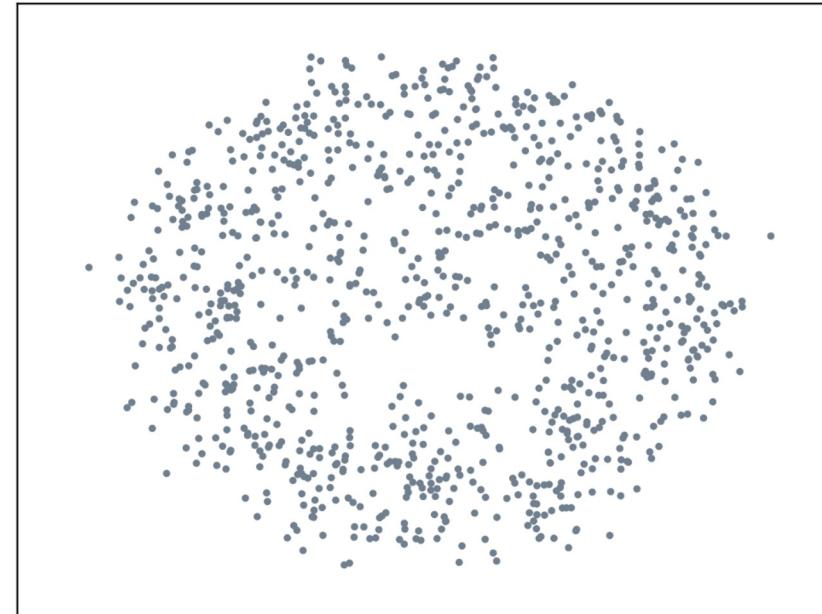
Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	AVG
ConSERT (Yan et al., 2021)	64.64	78.49	69.07	79.72	75.95	73.97	67.31	72.74
SimCSE (Gao et al., 2021)	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
DCLR (Zhou et al., 2022)	70.81	83.73	75.11	82.56	78.44	78.31	71.59	77.22
ESimCSE (Wu et al., 2021)	73.40	83.27	77.25	82.66	78.81	80.17	72.30	78.27
DiffCSE (Chuang et al., 2022)	72.28	84.43	76.47	83.90	80.54	80.59	71.23	78.49
PromptBERT (Jiang et al., 2022)	71.56	84.58	76.98	84.47	80.60	81.60	69.87	78.54
SNCSE (Wang et al., 2022)	70.67	84.79	76.99	83.69	80.51	81.35	74.77	78.97
RankEncoder	74.88	85.59	78.61	83.50	80.56	81.55	75.78	80.07

RankEncoder

- RankEncoder can be aware of the fine-grain interaction between the similar sentences in the corpus



(a) PromptBERT



(b) RankEncoder

RankEncoder

- Better uniformity

	Base Encoder E		
	SimCSE	PromptBERT	SNCSE
E	-2.42	-1.49	-2.21
RankEncoder $_E$	-3.23	-3.31	-3.20

Conclusion

- We are closing the gap between unsupervised and supervised sentence representations:

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
<i>Unsupervised models</i>								
* SimCSE-BERT _{base}	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
RankEncoder (BERT-base)	74.88	85.59	78.61	83.50	80.56	81.55	75.78	80.07
<i>Supervised models</i>								
* SimCSE-BERT _{base}	75.30	84.67	80.19	85.40	80.82	84.25	80.39	81.57
PromptBERT (BERT-base)	75.48	85.59	80.57	85.99	81.08	84.56	80.52	81.97

- Contrastive learning should have more potential in NLP for using pre-trained language models in representation learning!

Schedule

- 17:00 – 17:10 **Part 1** Introduction [Hung-yi]
- 17:10 – 17:40 **Part 2** Why do PLMs work [Hung-yi]
- 17:40 – 18:20 **Part 3** How to use PLMs: Contrastive Learning for PLMs [Yung-Sung]
- 18:20 – 18:30 Q&A for Part 1+2+3
- 18:30 – 18:40 Break
- 18:40 – 19:05 **Part 4** How to use PLMs: Parameter-efficient fine-tuning [Cheng-Han]
- 19:05 – 19:50 **Part 5** How to use PLMs: Using PLMs with different amounts of data [Cheng-Han]
- 19:50 – 20:00 Conclusion and Future work + Q&A



2022 AACL-IJCNLP

Part 4: How to use PLMs: Parameter-efficient fine-tuning

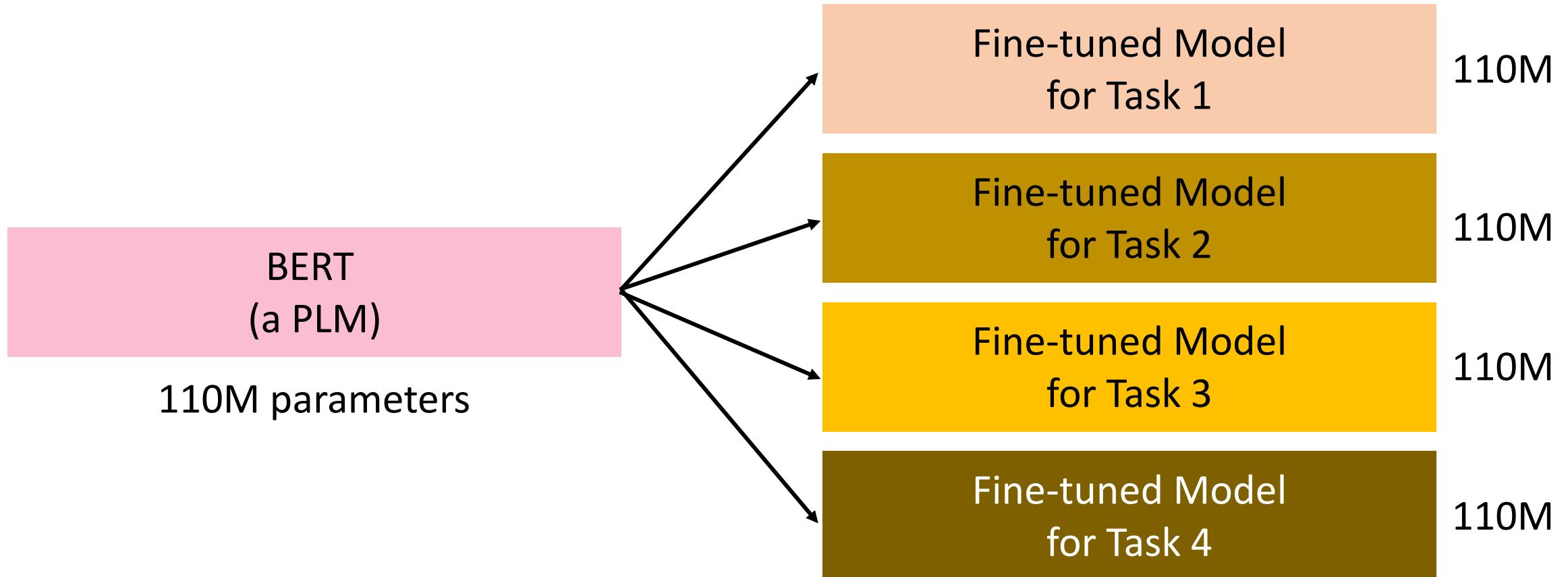
Cheng-Han Chiang

National Taiwan University



Parameter-Efficient Fine-tuning

- PLMs are gigantic
 - Need a copy for each downstream task

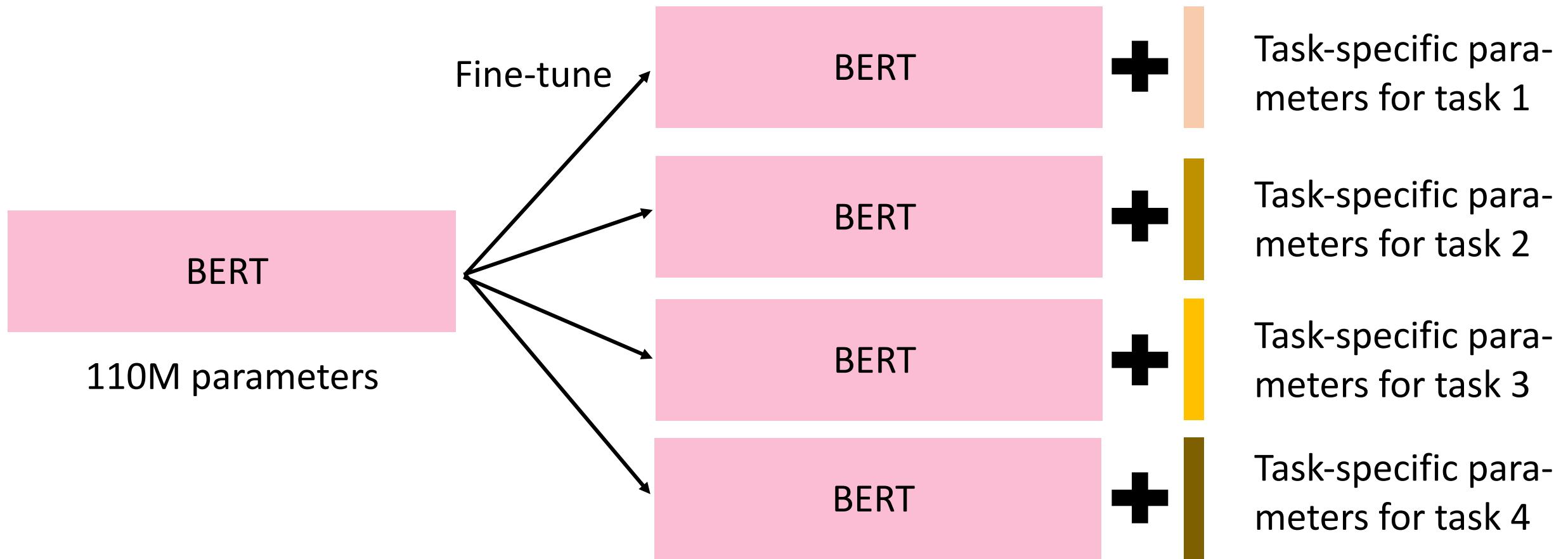


Parameter-Efficient Fine-tuning

- 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

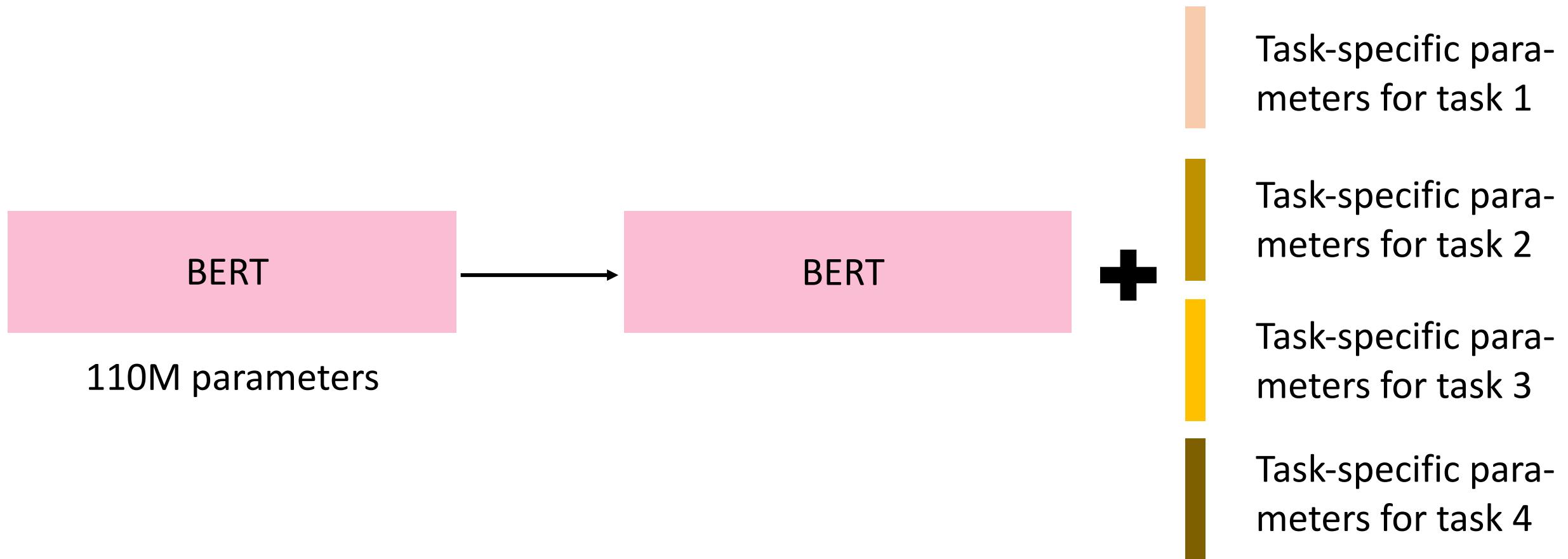
Parameter-Efficient Fine-tuning

- Use a small amount of parameters for each downstream task



Parameter-Efficient Fine-tuning

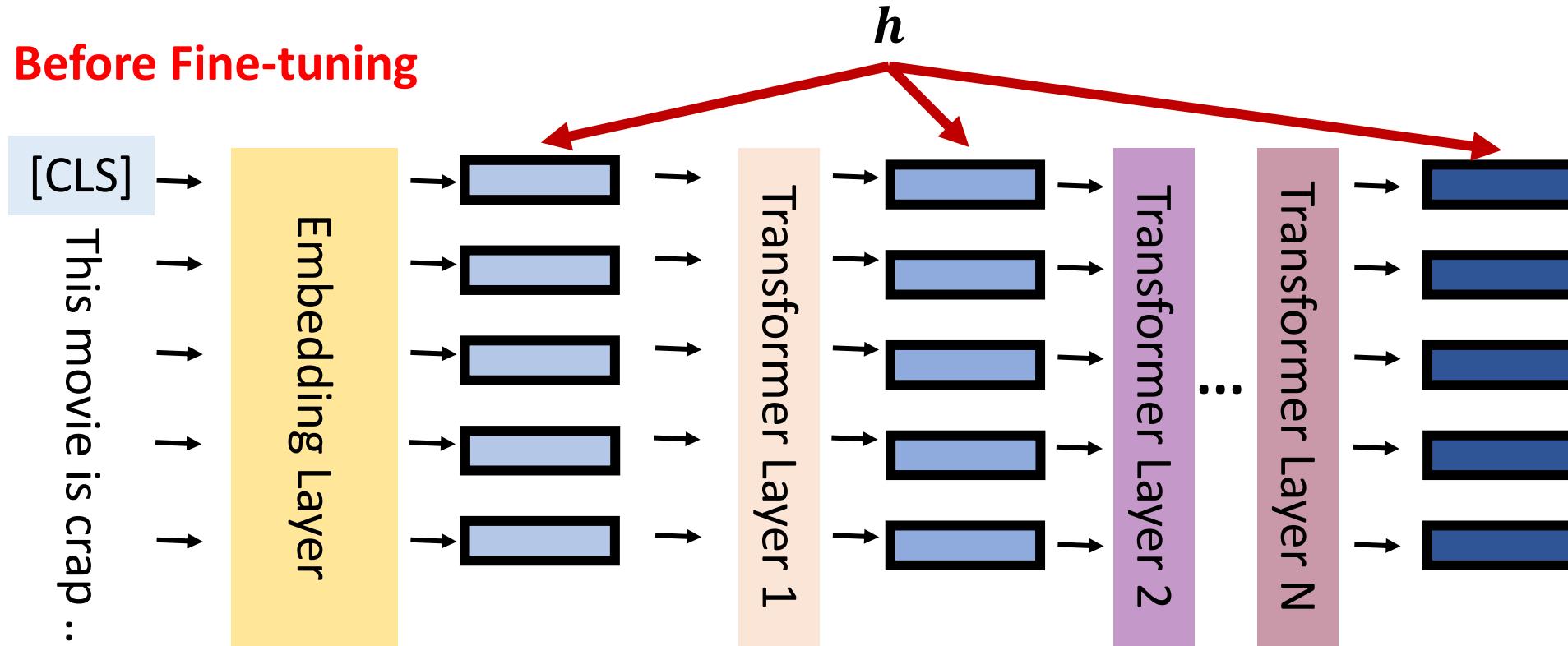
- Use a small amount of parameters for each downstream task



Parameter-Efficient Fine-tuning

- What is standard fine-tuning really doing?
 - Modify the hidden representations (h) of the PLM such that it can perform well on downstream task

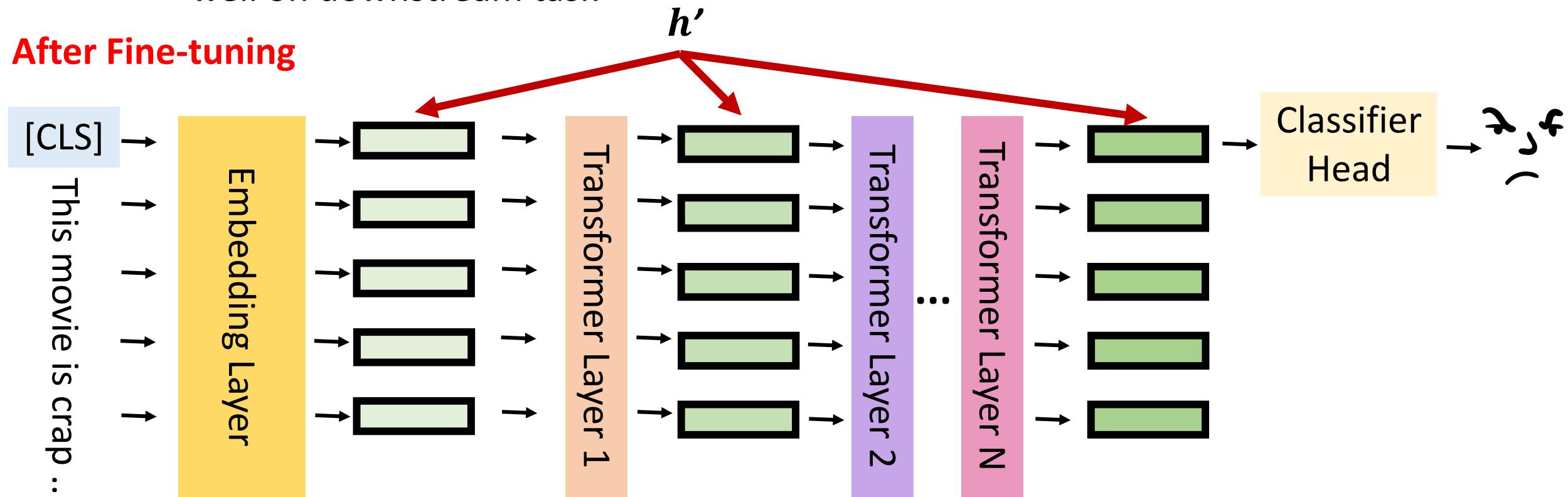
Before Fine-tuning



Parameter-Efficient Fine-tuning

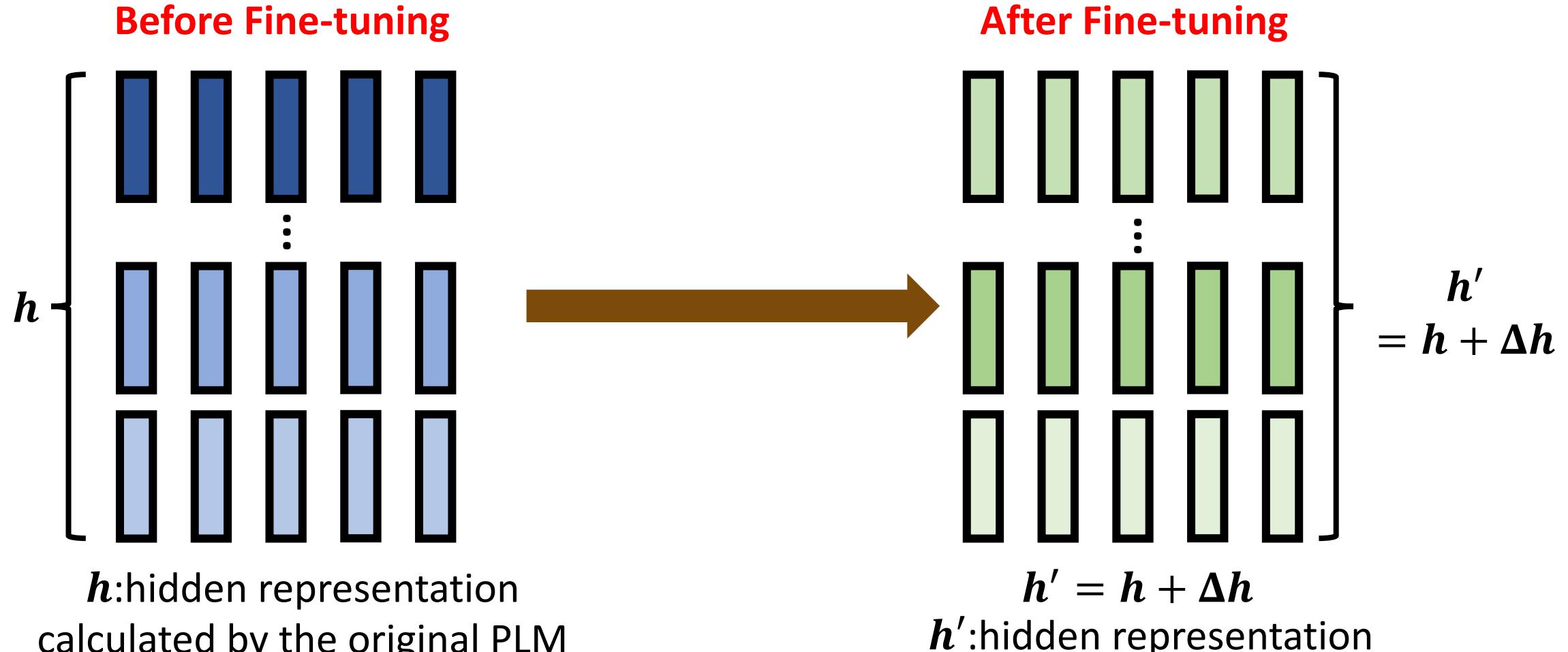
- What is standard fine-tuning really doing?
 - Modify the hidden representations (h) of the PLM such that it can perform well on downstream task

After Fine-tuning



Parameter-Efficient Fine-tuning

- Fine-tuning = modifying the hidden representation based on a PLM

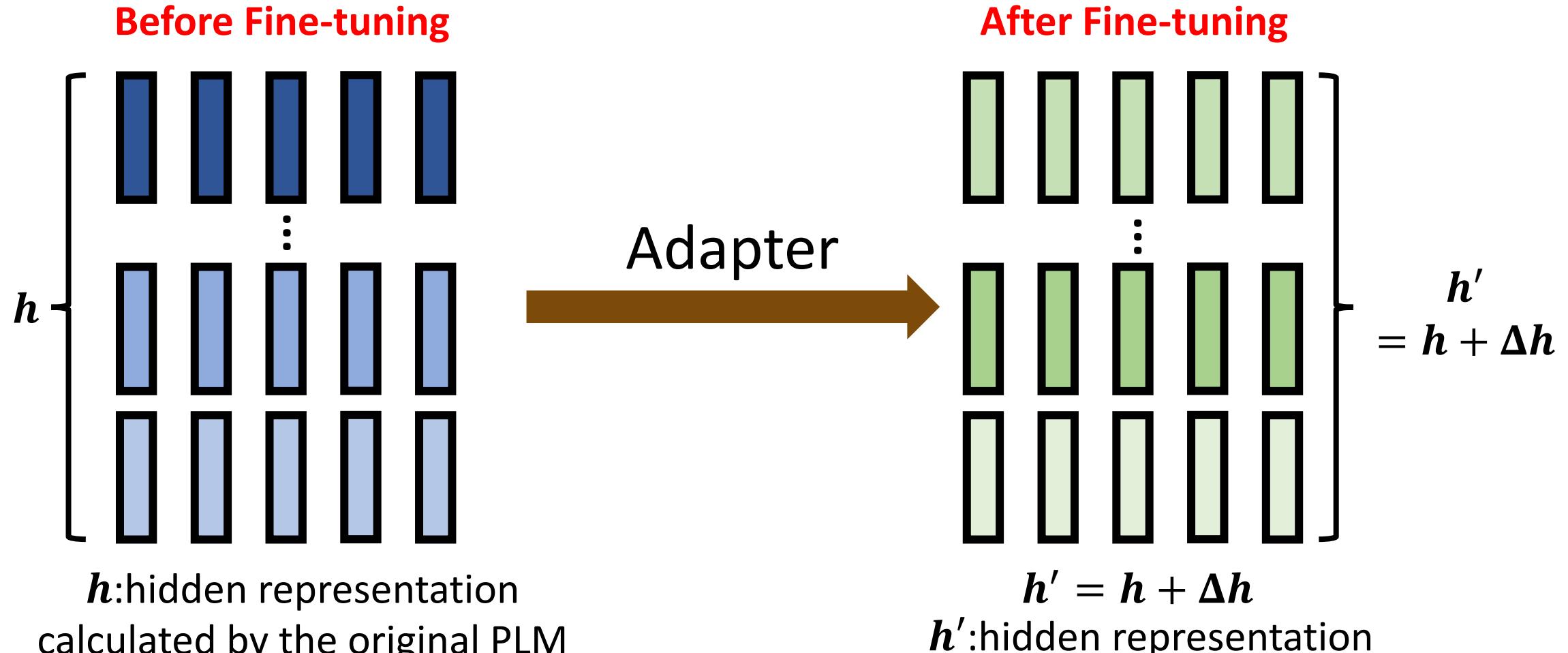




Part 4:
How to use PLMs:
Parameter-efficient fine-tuning
4-1 Adapter

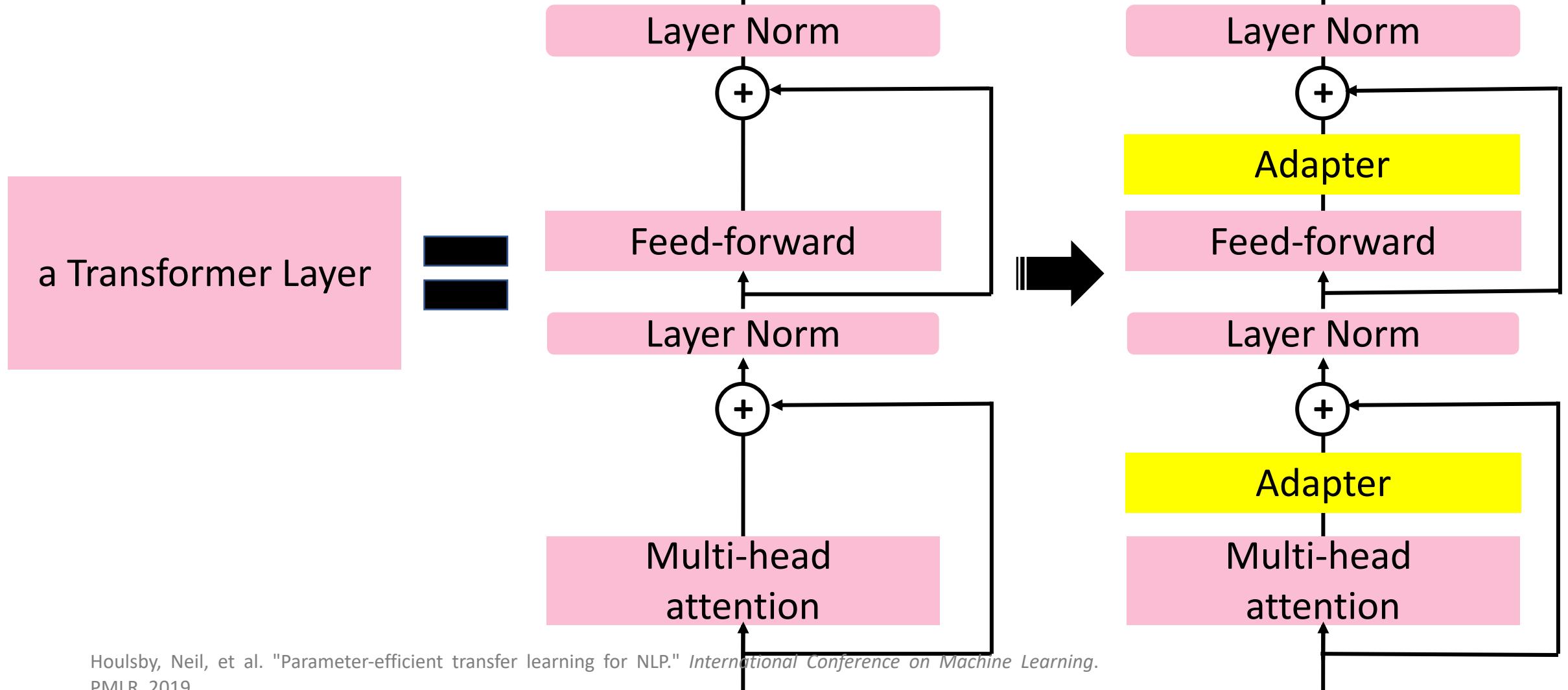
Parameter-Efficient Fine-tuning: Adapter

- Use special submodules to modify hidden representations!



Parameter-Efficient Fine-tuning: Adapter

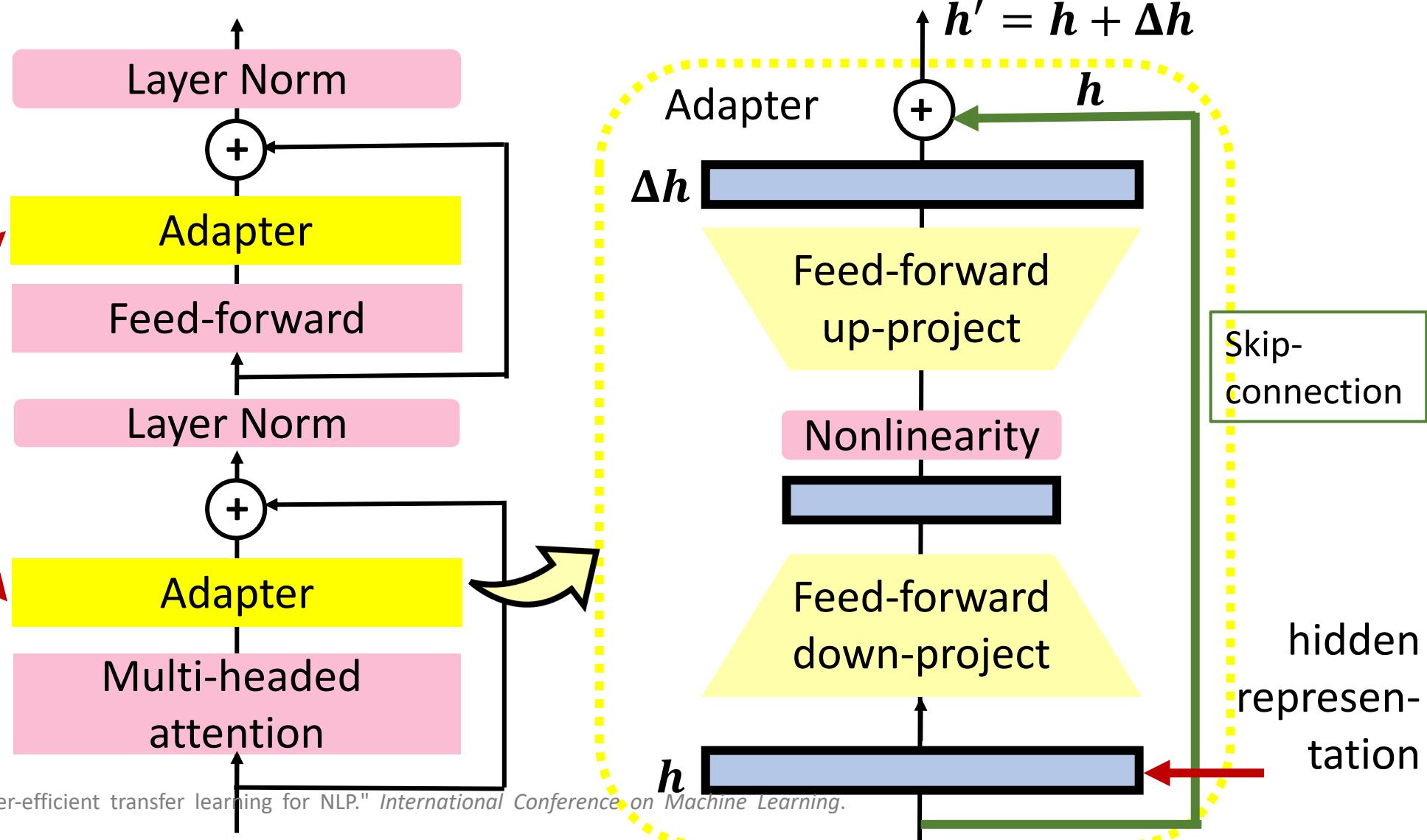
- Adapters: small trainable submodules inserted in transformers



Parameter-Efficient Fine-tuning: Adapter

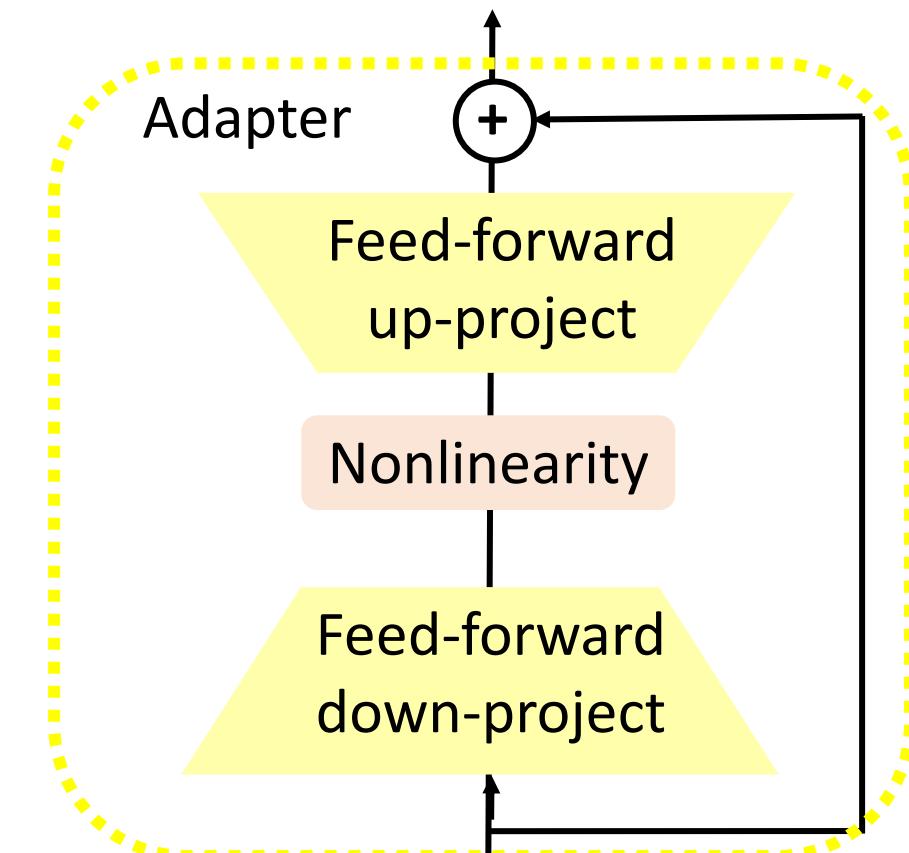
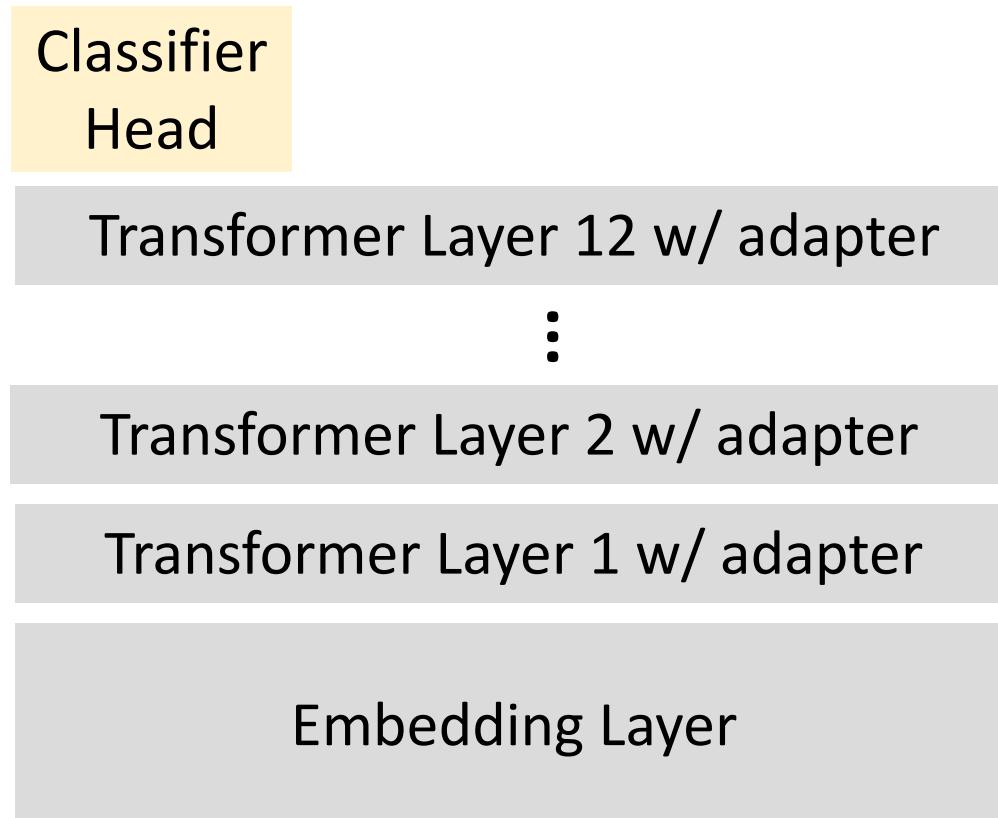
- Adapters

Inside of the transformer layer, only adapters are updated



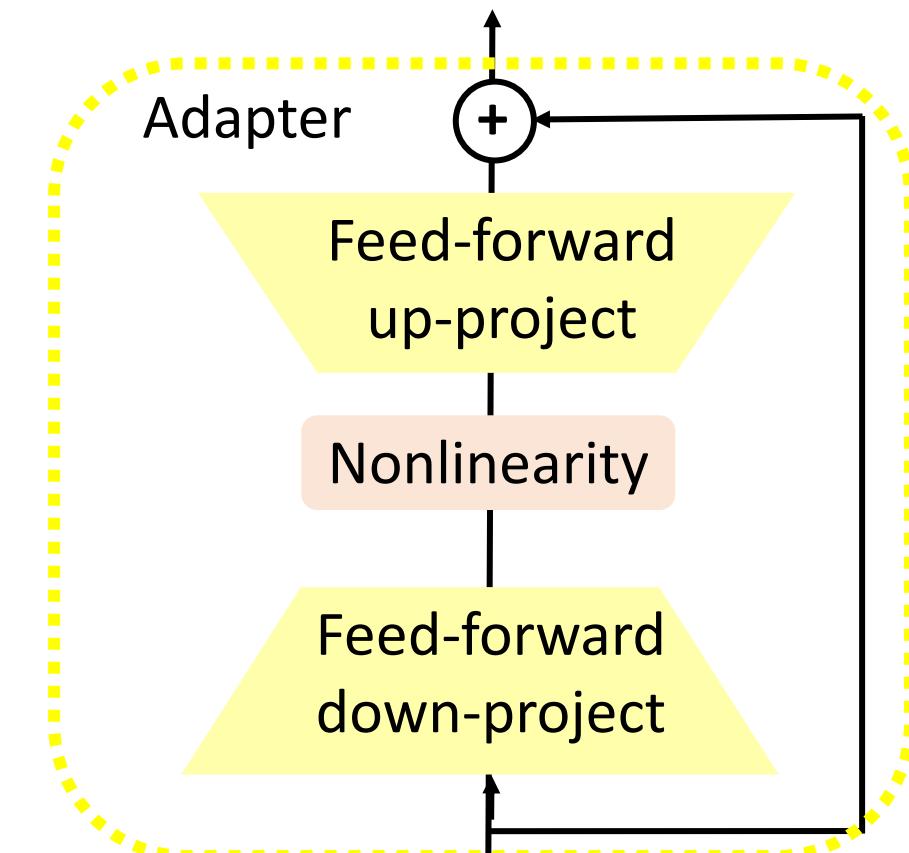
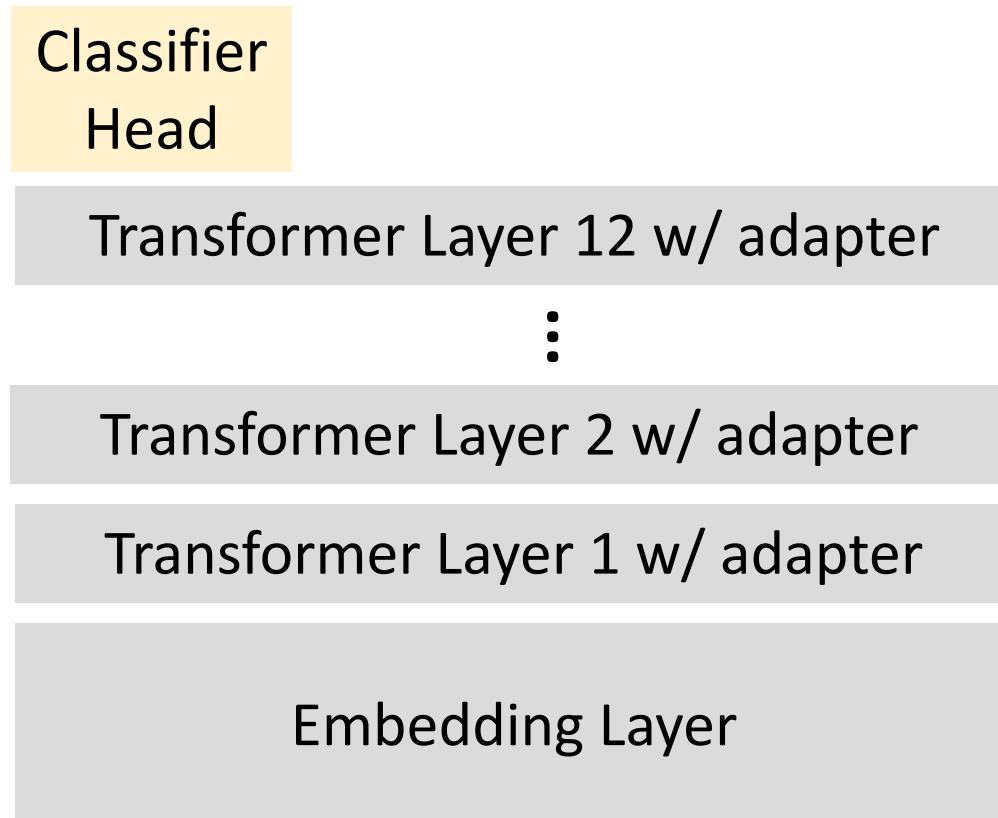
Parameter-Efficient Fine-tuning: Adapter

- Adapters: During fine-tuning, only update the adapters and the classifier head



Parameter-Efficient Fine-tuning: Adapter

- Adapters: All downstream tasks share the PLM; the adapters in each layer and the classifier heads are the task-specific modules



Part 4:

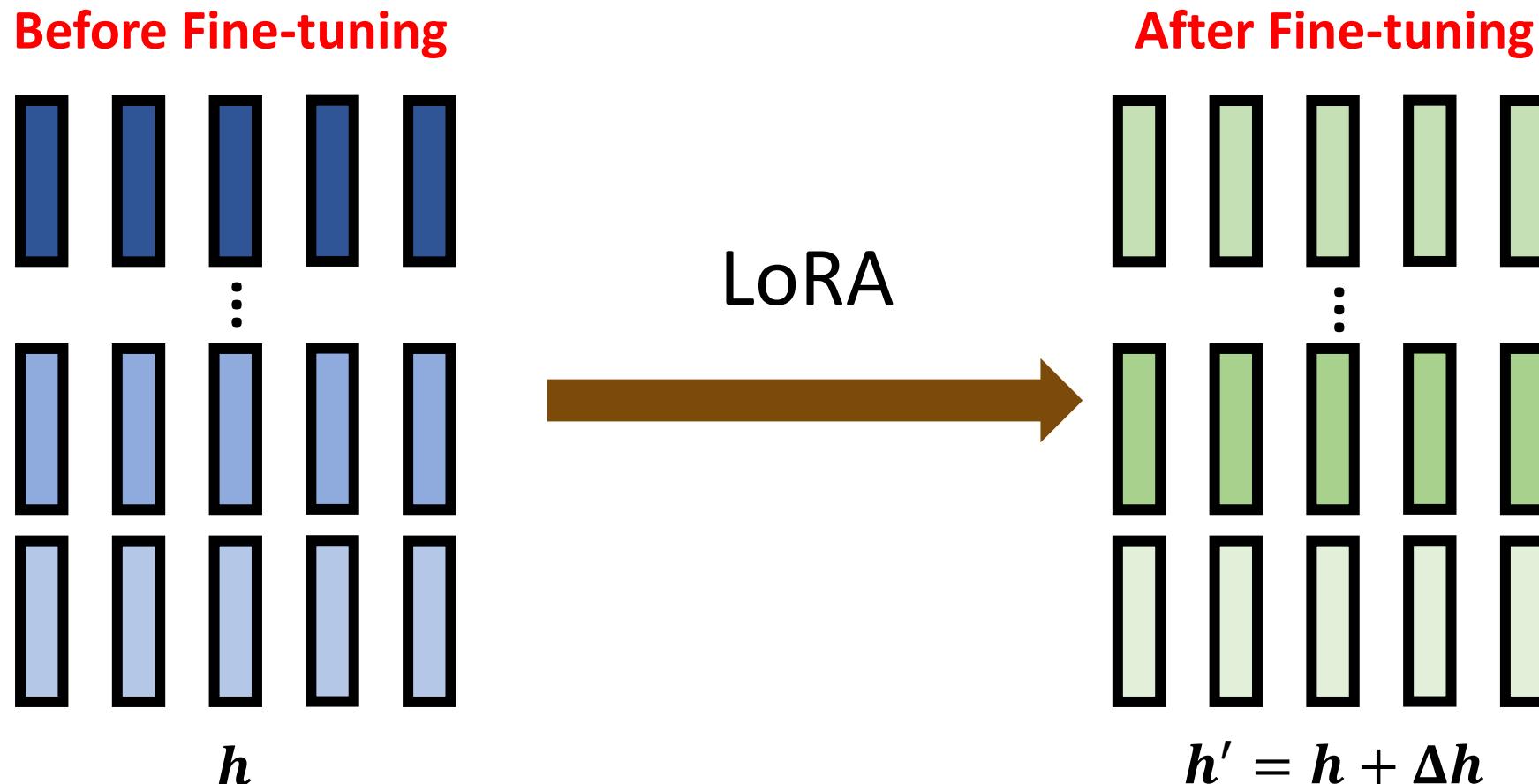
How do PLMs work:

Parameter-efficient fine-tuning

4-2 LoRA

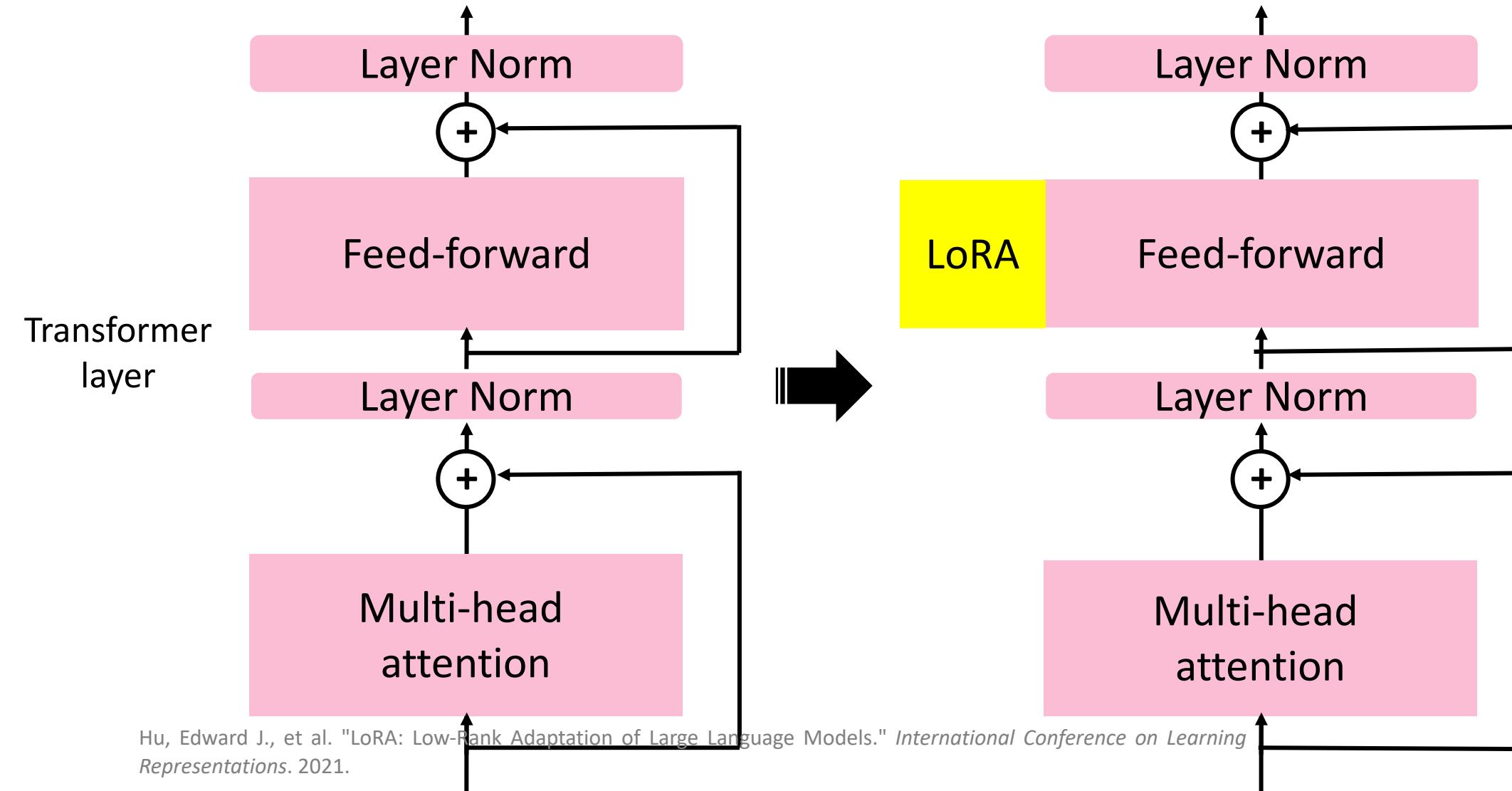
Parameter-Efficient Fine-tuning: LoRA

- Use special submodules to modify hidden representations!



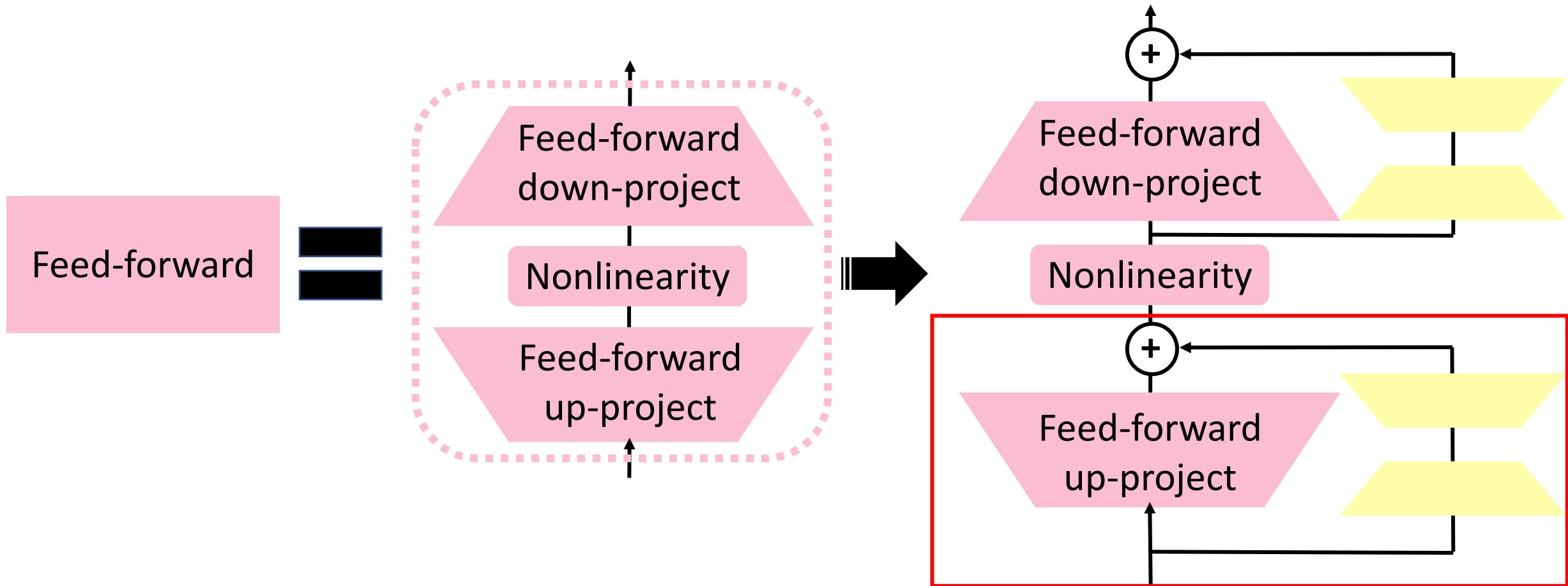
Parameter-Efficient Fine-tuning: LoRA

- LoRA: Low-Rank Adaptation of Large Language Models



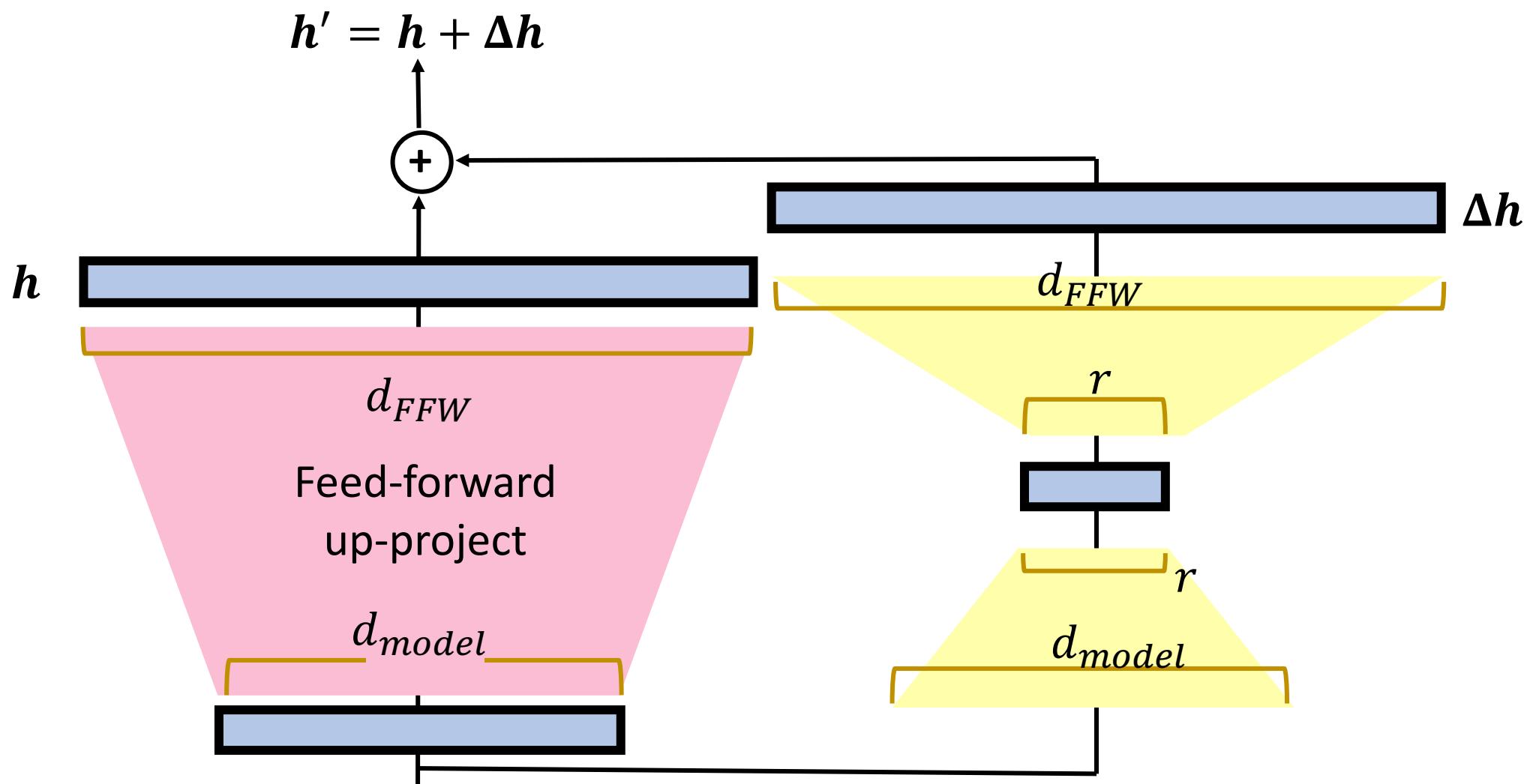
Parameter-Efficient Fine-tuning: LoRA

- LoRA



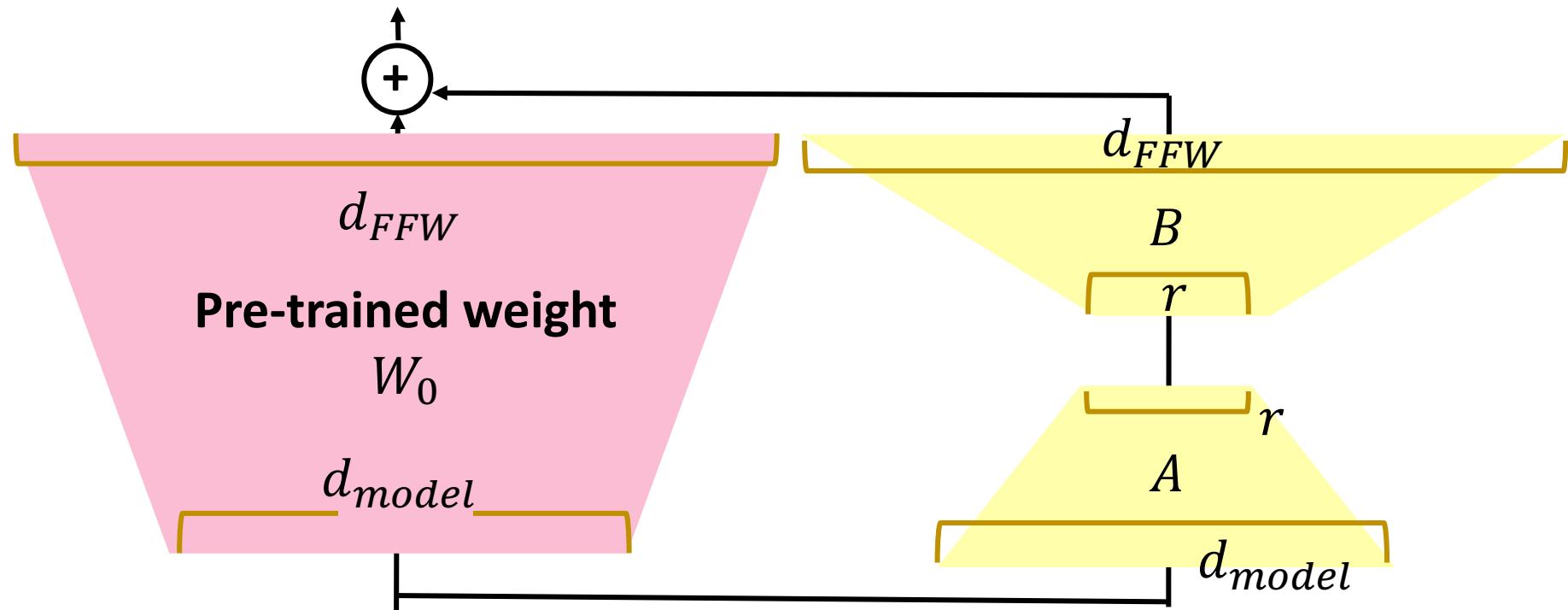
Parameter-Efficient Fine-tuning: LoRA

- LoRA



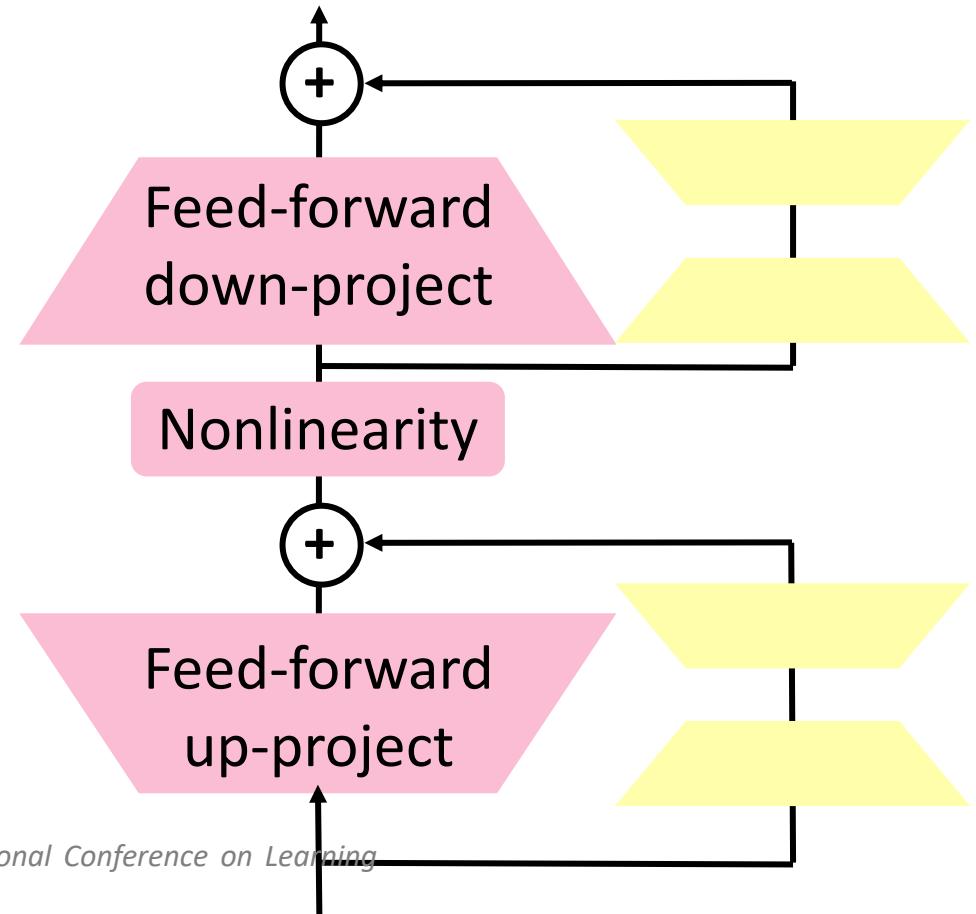
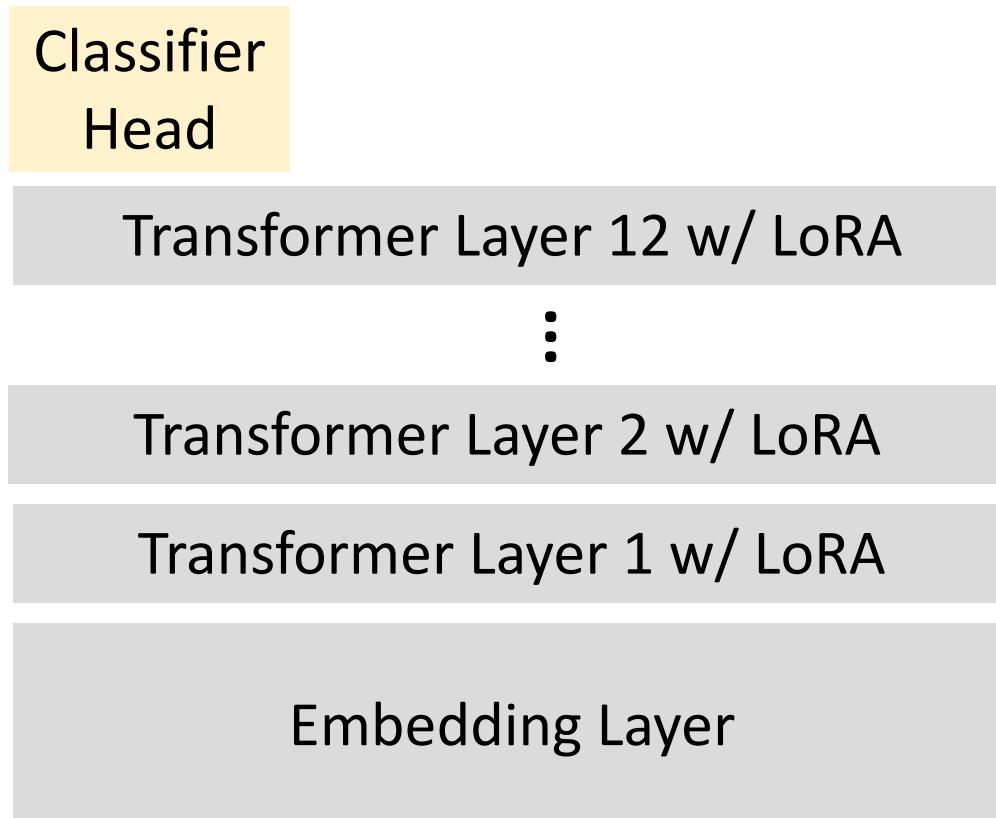
Parameter-Efficient Fine-tuning: LoRA

- **Low-Rank Adaptation** of Large Language Models
- Motivation: Downstream fine-tunings have low intrinsic dimension
- Weight after fine-tuning = W_0 (pre-trained weight) + ΔW (updates to the weight)
- Hypothesis: The updates to the weight (ΔW) also gave a low intrinsic rank
- Fine-tuned weight = $W_0 + \Delta W = W_0 + BA$, rank $r \ll \min(d_{FFW}, d_{model})$



Parameter-Efficient Fine-tuning: LoRA

- LoRA: All downstream tasks share the PLM; the LoRA in each layer and the classifier heads are the task-specific modules

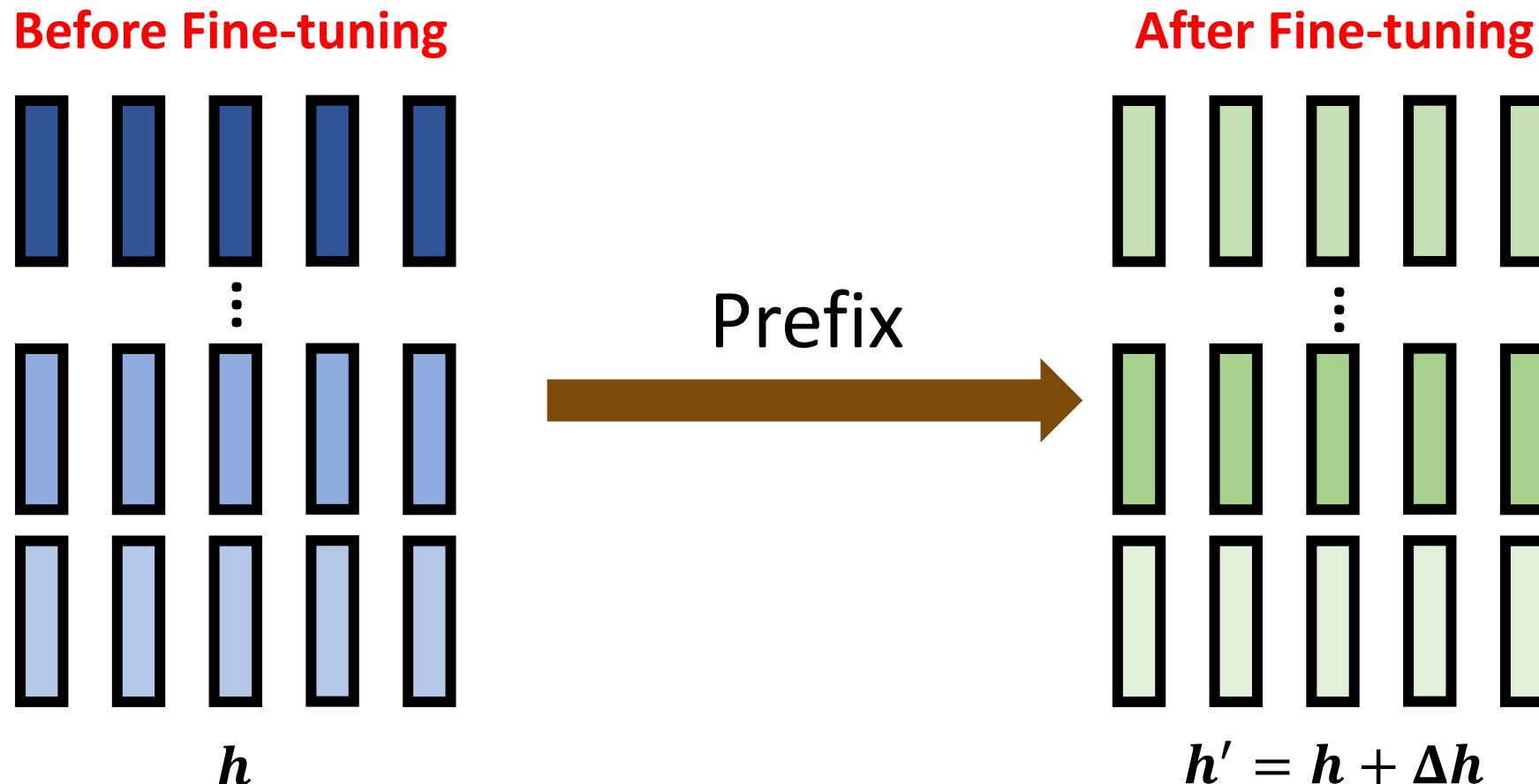




Part 4:
How do PLMs work:
Parameter-efficient fine-tuning
4-3 Prefix tuning

Parameter-Efficient Fine-tuning: Prefix Tuning

- Use special submodules to modify hidden representations!



Parameter-Efficient Fine-tuning: Prefix Tuning

- What is “*prefix*”

prefix noun [C]

UK /'pri:.fiks/ US /'pri:.fiks/

prefix noun [C] (GRAMMAR)

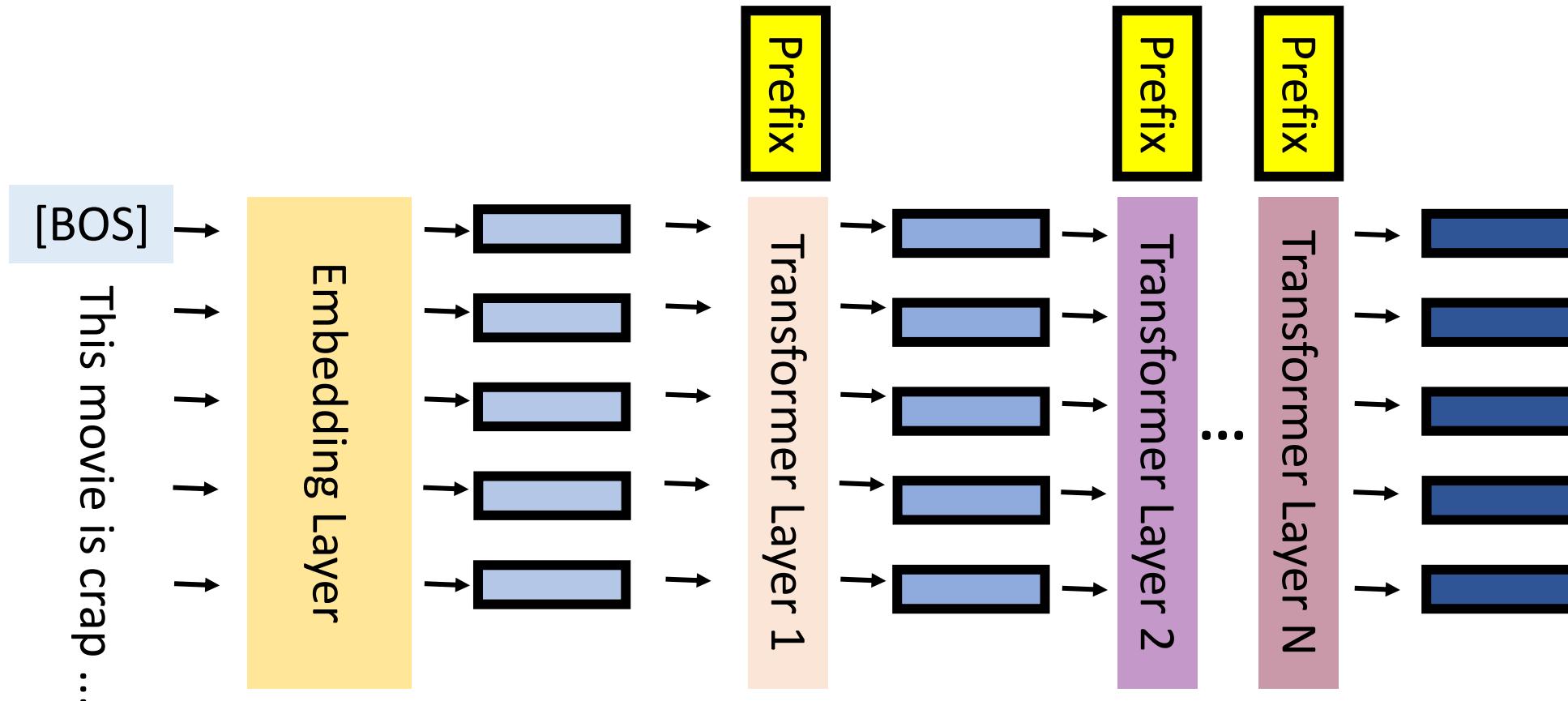
B2 LANGUAGE

a letter or group of letters added to the beginning of a word to make a new word:

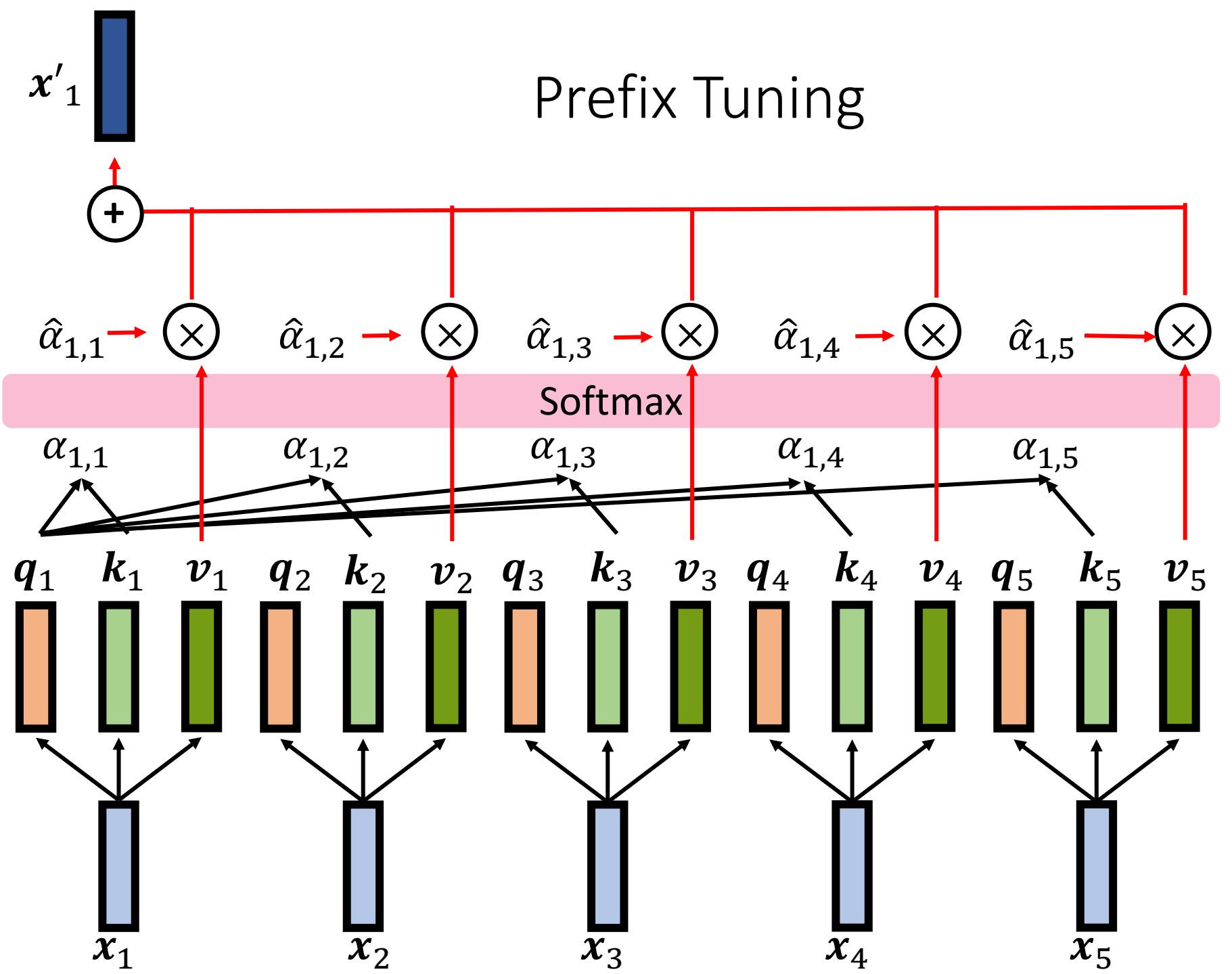
- Something that is put in front of another something

Parameter-Efficient Fine-tuning: Prefix Tuning

- Prefix Tuning: Insert trainable prefix in each layer



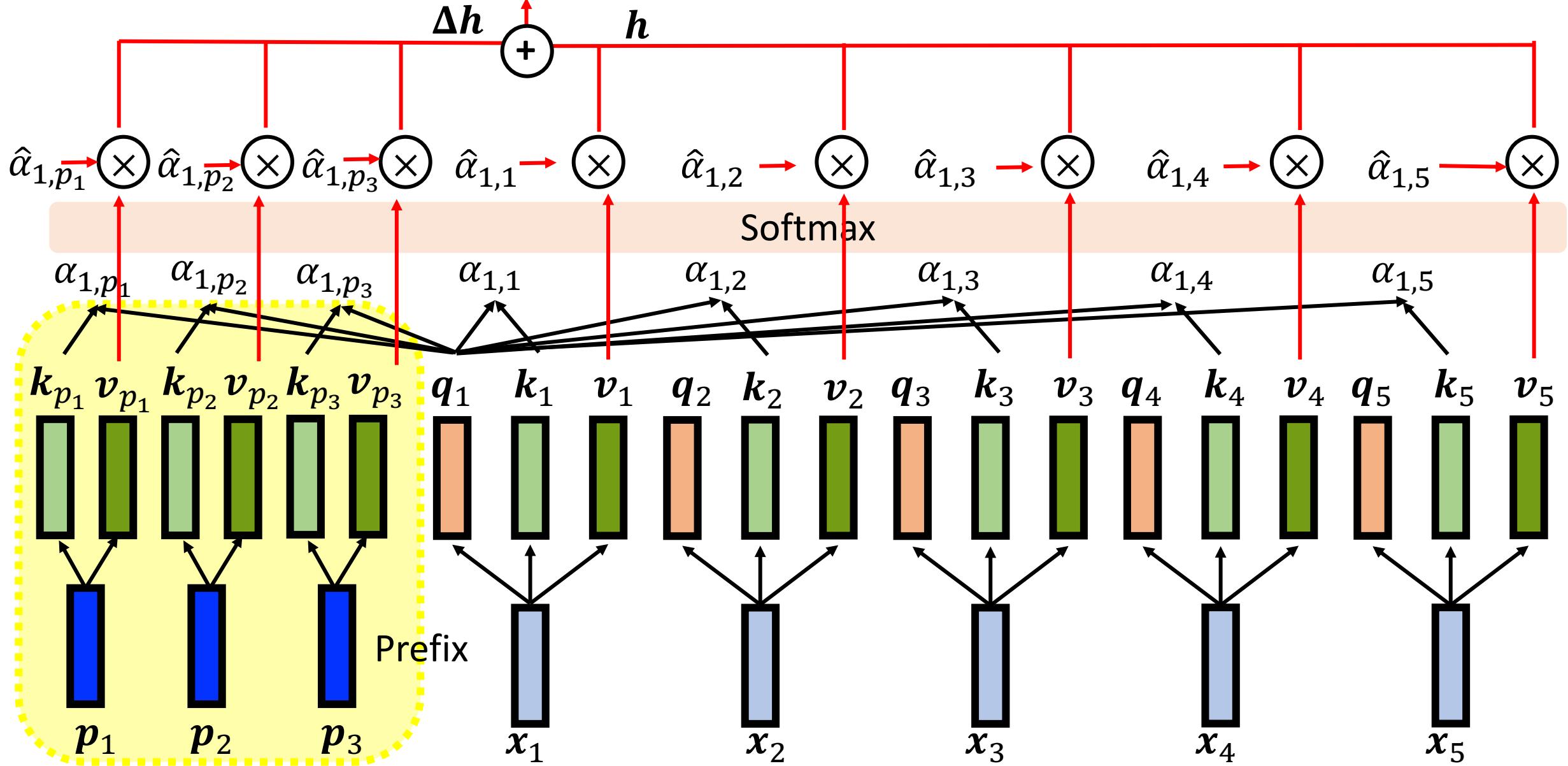
Standard
Self-Attention



$$h' = h + \Delta h$$

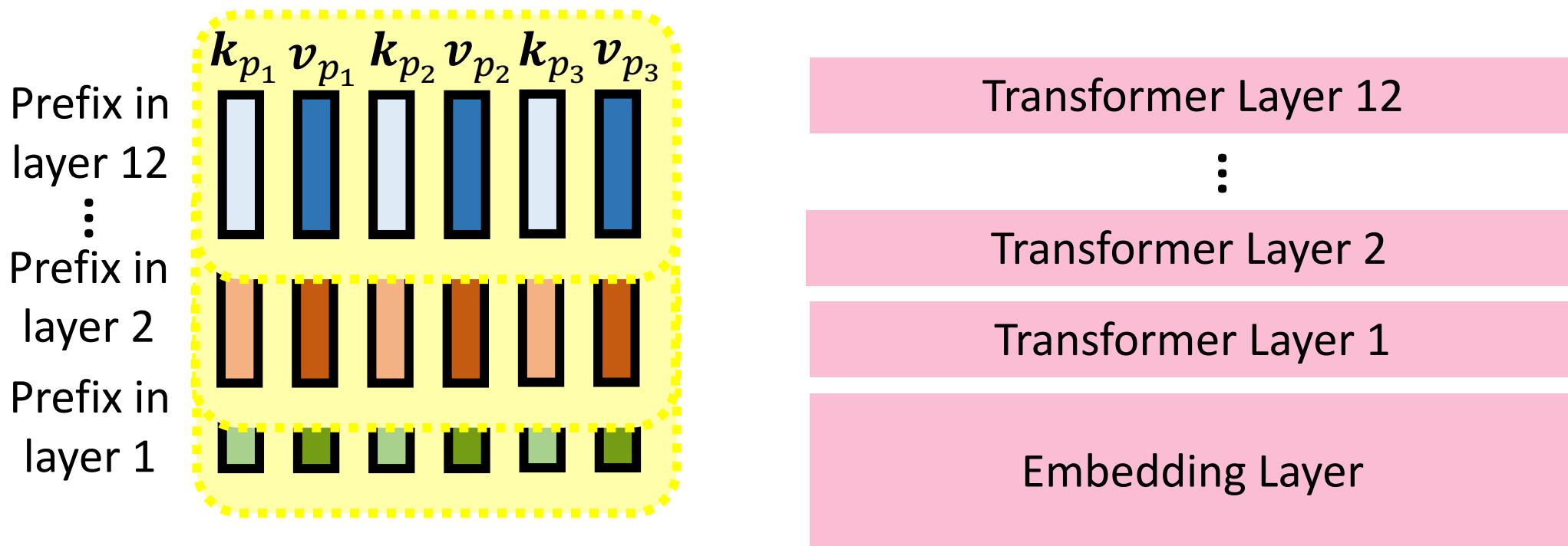


Prefix Tuning



Parameter-Efficient Fine-tuning: Prefix Tuning

- Prefix Tuning: Only the prefix (key and value) are updated during fine-tuning

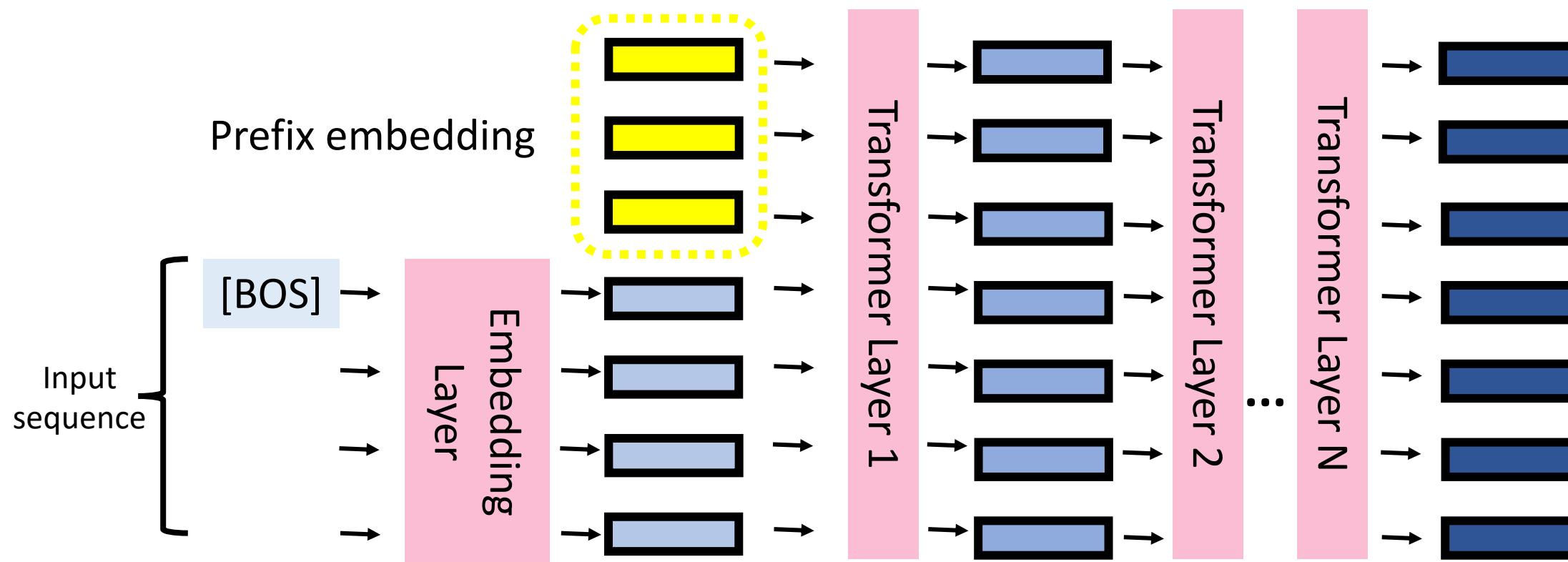




Part 4:
How do PLMs work:
Parameter-efficient fine-tuning
4-4 (Soft) Prompt tuning

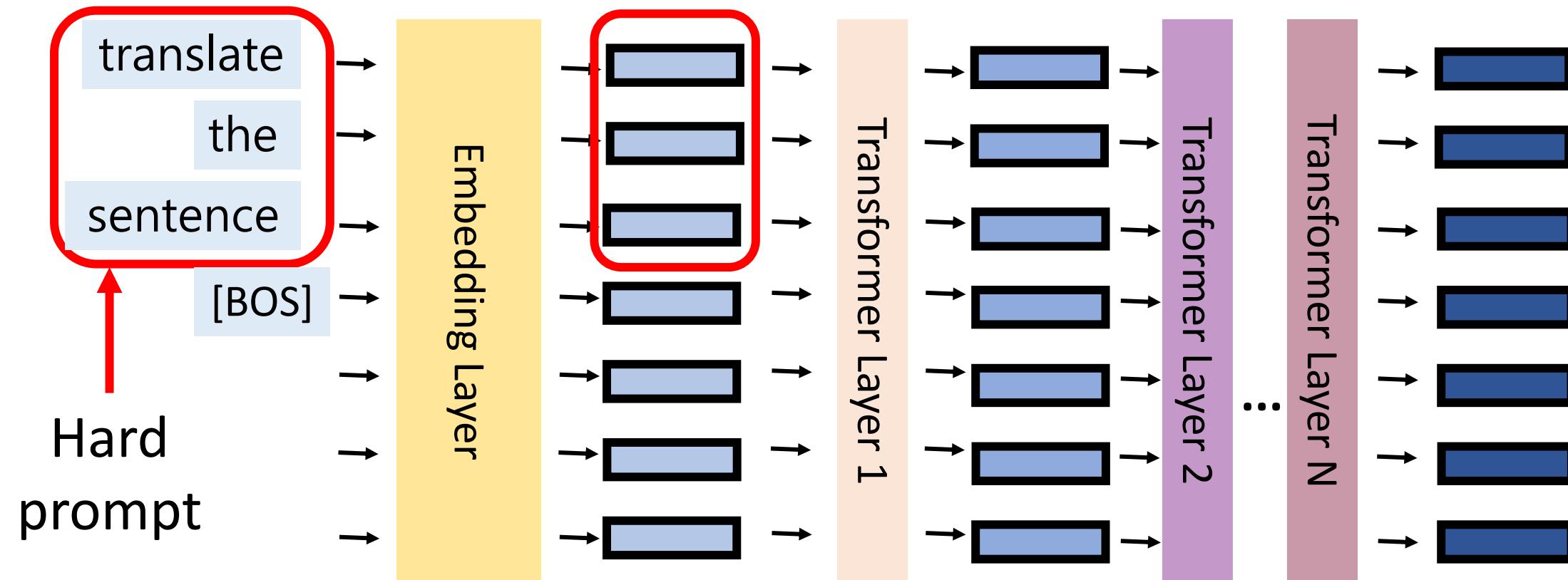
Parameter-Efficient Fine-tuning: Soft Prompting

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



Parameter-Efficient Fine-tuning: Soft Prompting

- Soft Prompting can be considered as the *soften* version of prompting
 - (Hard) prompting: add words in the input sentence

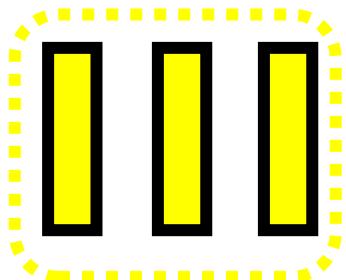


Parameter-Efficient Fine-tuning: Soft Prompting

- Hard Prompts: words (that are originally in the vocabulary)

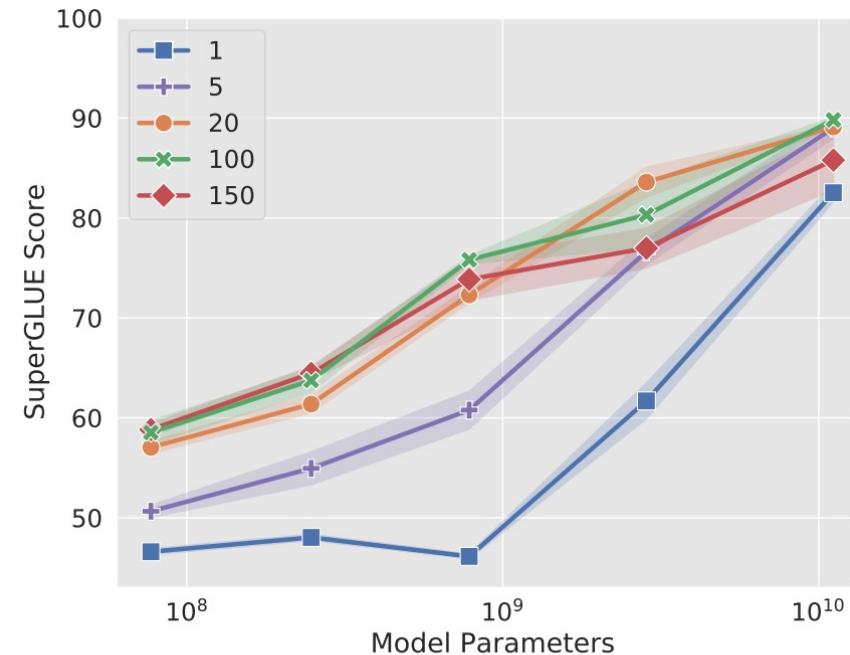
Translate the sentence

- Soft Prompts: vectors (can be initialized from some word embeddings)



Parameter-Efficient Fine-tuning: Soft Prompting

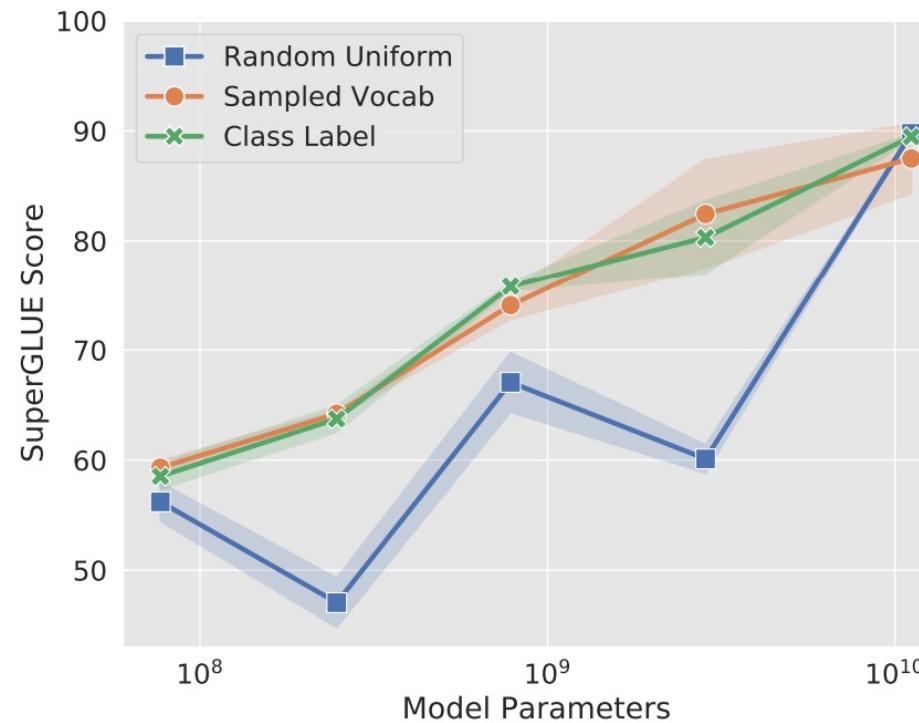
- How to determine the length of the soft prompt embedding
 - The prompt needs to be long enough
 - Increasing the prompt length shows diminishing performance gain when the length is long enough



(a) Prompt length

Parameter-Efficient Fine-tuning: Soft Prompting

- How to initialize the soft prompt embedding?
 - Random initialization
 - Sample from the word embedding of top 5000 frequent words
 - Class label in the downstream task



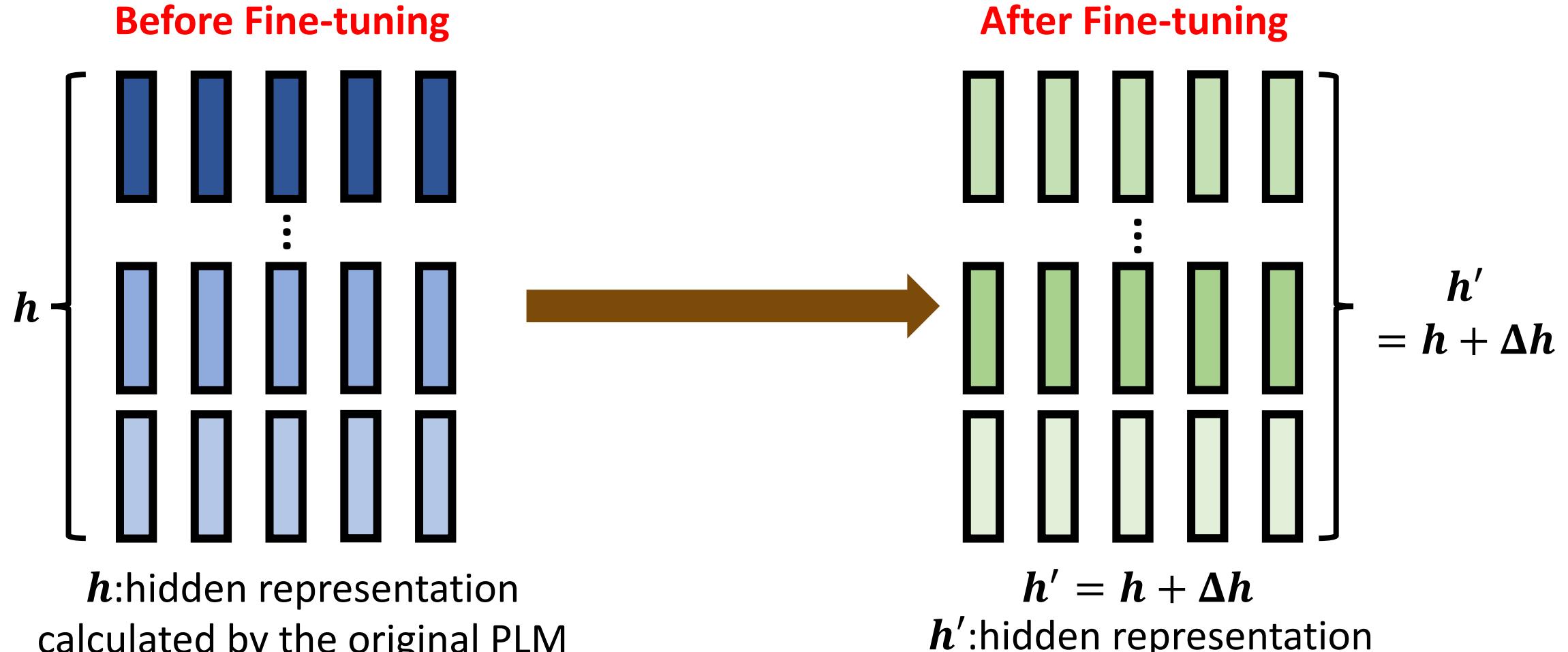
(b) Prompt initialization



Part 4:
How do PLMs work:
Parameter-efficient fine-tuning
4-5 Short summary

Parameter-Efficient Fine-tuning

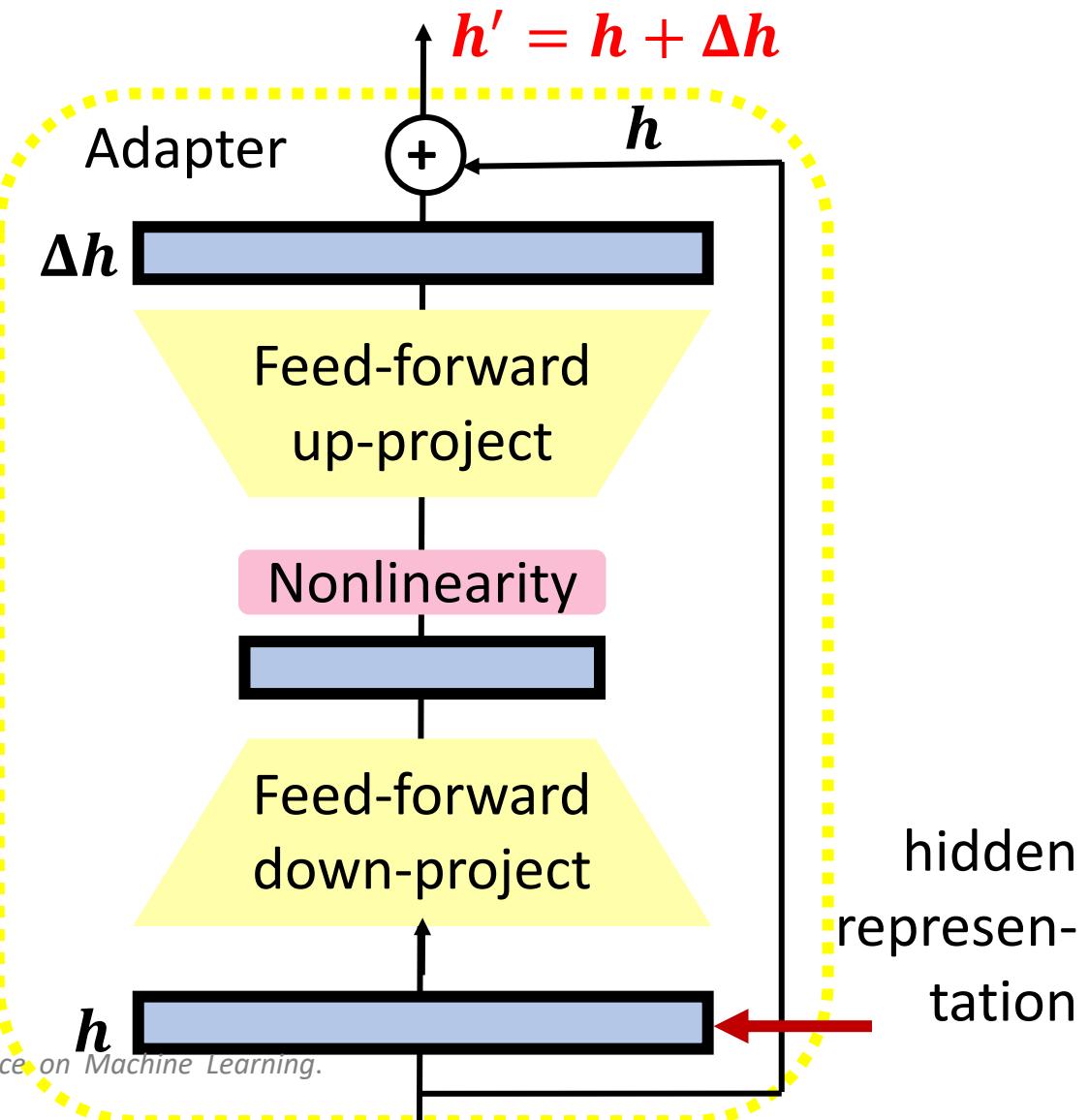
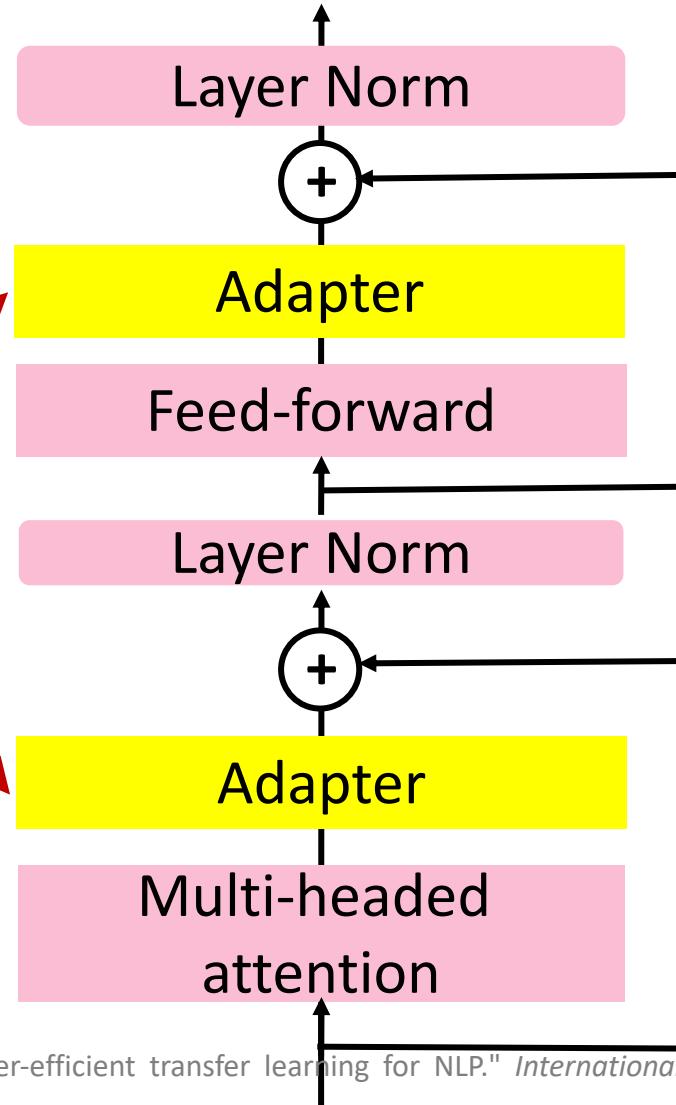
- Fine-tuning = modifying the hidden representation based on a PLM



Parameter-Efficient Fine-tuning: Adapter

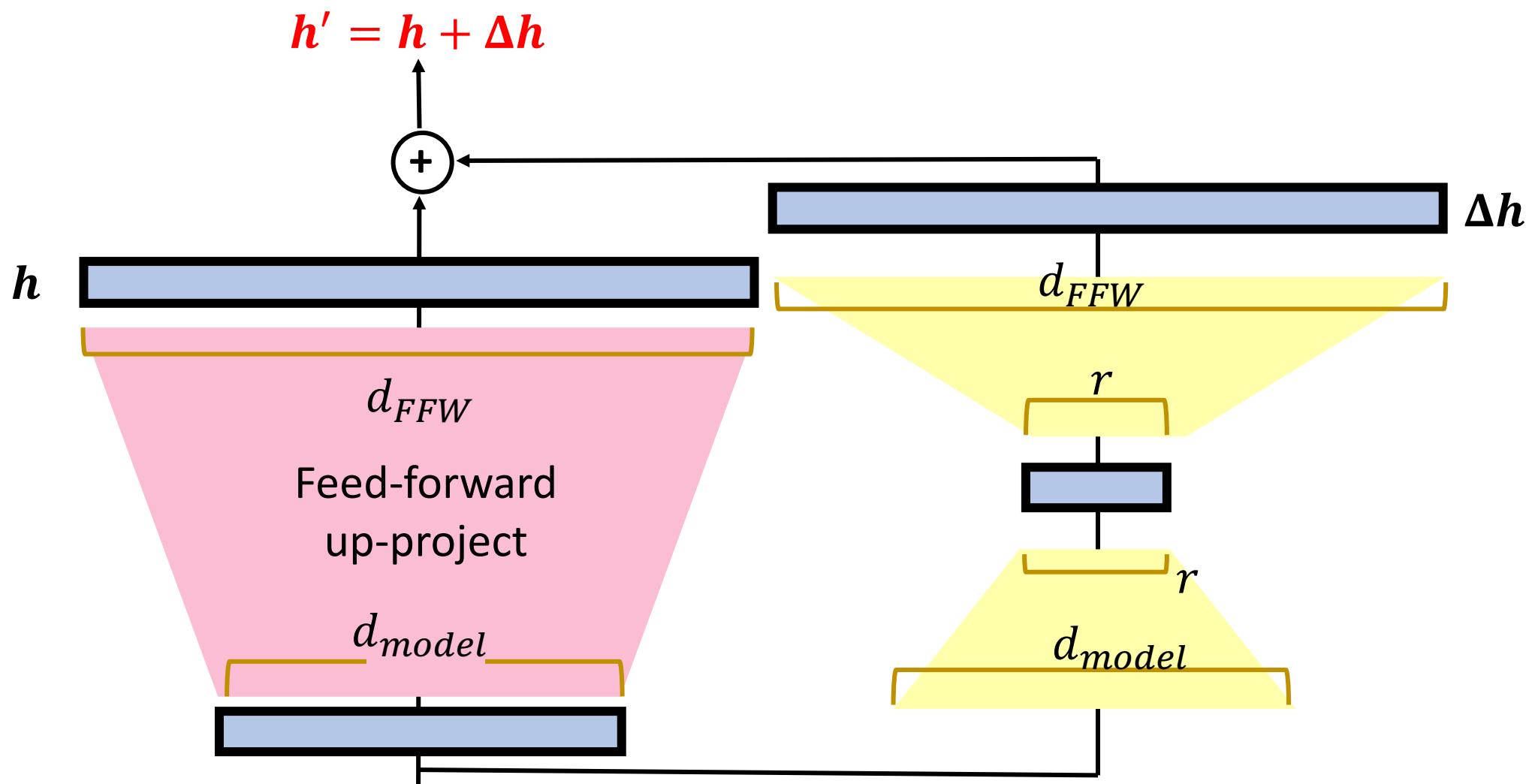
- Adapters

Inside of the transformer layer, only adapters are updated



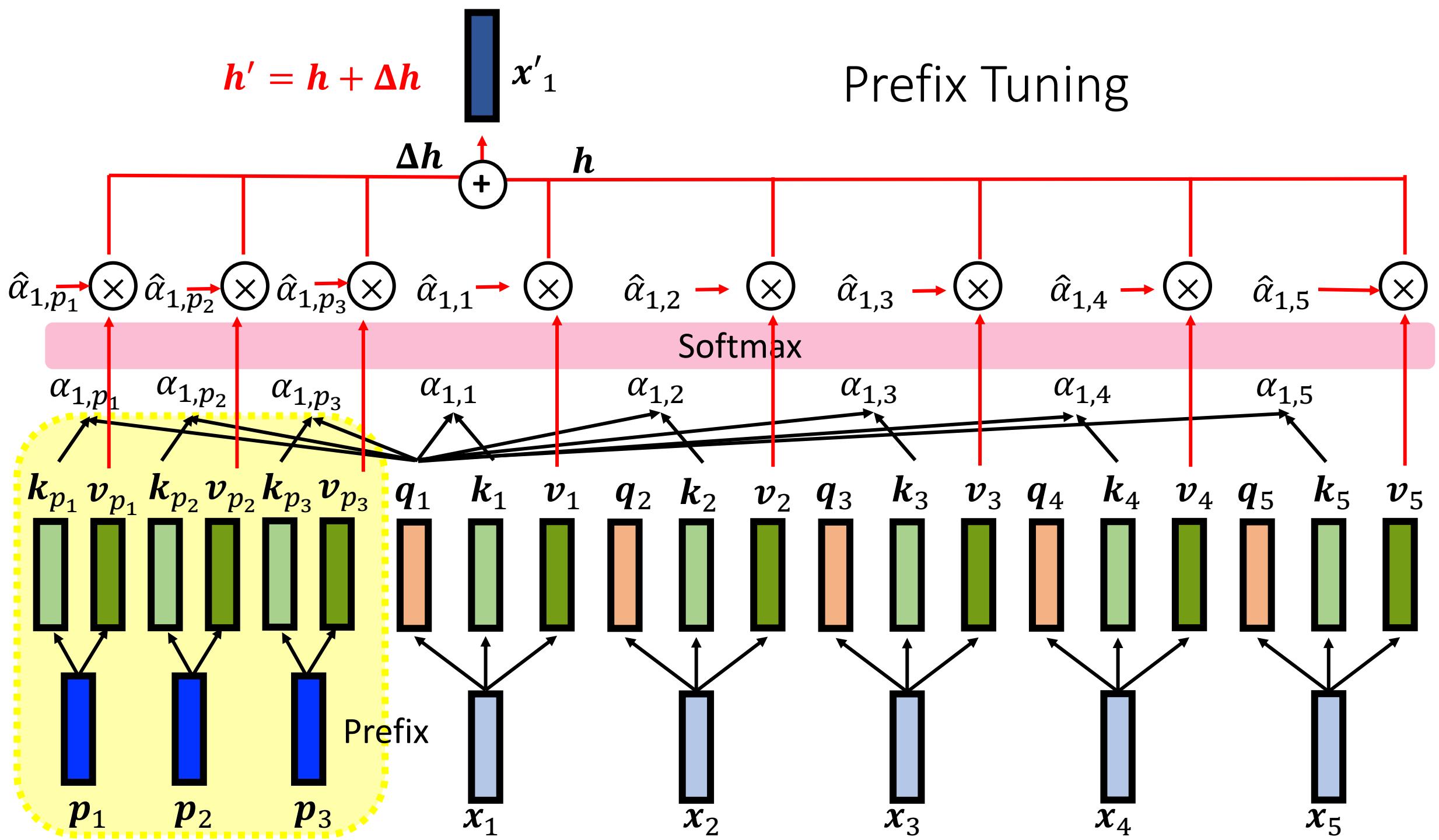
Parameter-Efficient Fine-tuning: LoRA

- LoRA



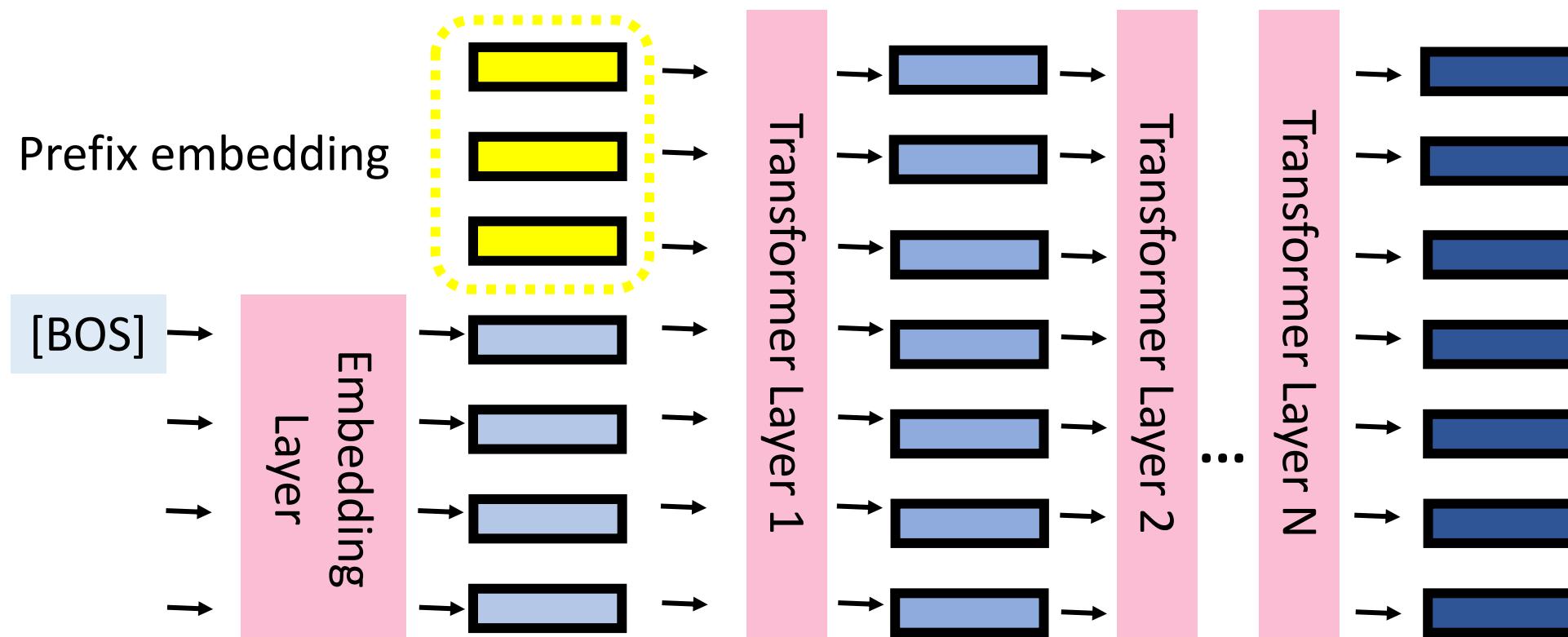
$$h' = h + \Delta h$$

Prefix Tuning



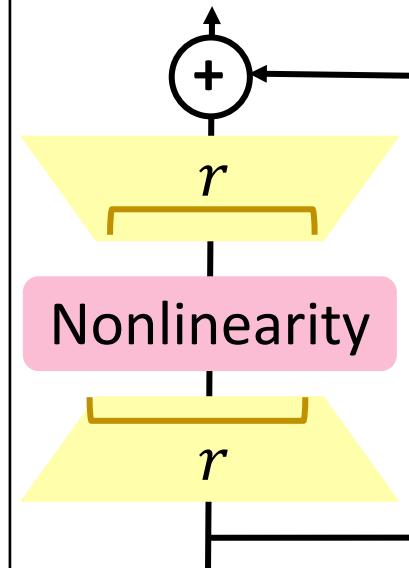
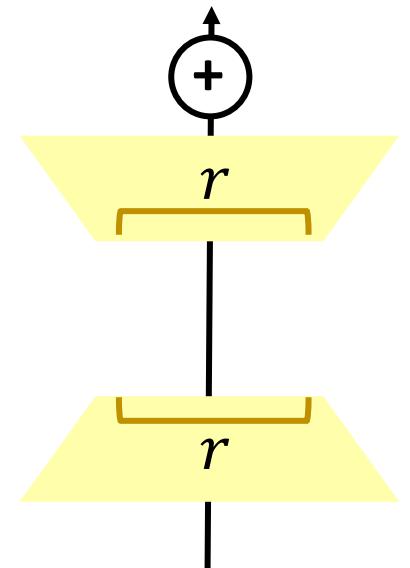
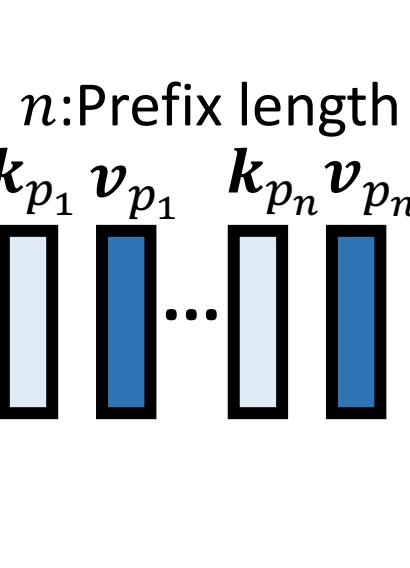
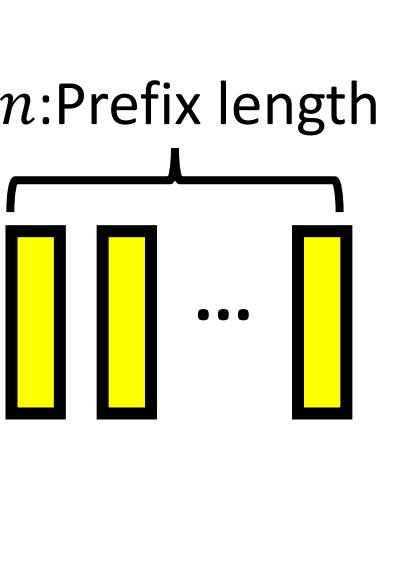
Parameter-Efficient Fine-tuning: Soft Prompting

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



Parameter-Efficient Fine-tuning

- Benefit 1: Drastically decreases the task-specific parameters

	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			n : Prefix length $k_{p_1}, v_{p_1}, k_{p_n}, v_{p_n}$... 	n : Prefix length ... 

*not including the classifier head

Parameter-Efficient Fine-tuning

- Benefit 2: Less easier to overfit on training data; better out-of-domain performance

	Dataset	Domain	Standard	Soft Prompt	$\Delta = \text{Soft Prompt} - \text{Standard}$
Training dataset	SQuAD	Wiki	94.9 ± 0.2	94.8 ± 0.1	-0.1
	TextbookQA	Book	54.3 ± 3.7	66.8 ± 2.9	+12.5
OOD test dataset	BioASQ	Bio	77.9 ± 0.4	79.1 ± 0.3	+1.2
	RACE	Exam	59.8 ± 0.6	60.7 ± 0.5	+0.9
	RE	Wiki	88.4 ± 0.1	88.8 ± 0.2	+0.4
	DuoRC	Movie	68.9 ± 0.7	67.7 ± 1.1	-1.2
	DROP	Wiki	68.9 ± 1.7	67.1 ± 1.9	-1.8

Parameter-Efficient Fine-tuning

- Benefit 3: Fewer parameters to fine-tune, making them good candidates when training with small dataset

Dataset (Train set size)	low-resource				high-resource			
	CHEMPROT (4169)	ACL-ARC (1688)	SCIERC (3219)	HYP. (515)	RCT (180k)	AGNEWS (115k)	HELPFUL. (115k)	IMDB (20k)
RoBERTa-full model	81.7 _{0.8}	65.0 _{3.6}	78.5 _{1.8}	88.9 _{3.3}	87.0 _{0.1}	93.7 _{0.2}	69.1 _{0.6}	95.2 _{0.1}
RoBERTa-adapter, r=256	82.9 _{0.6}	67.5 _{4.3}	80.8 _{0.7}	90.4 _{4.2}	87.1 _{0.1}	93.8 _{0.1}	69.0 _{0.4}	95.7 _{0.1}

Parameter-Efficient Fine-tuning

- Which parameter-efficient fine-tuning should one select?
 - No one-size-fit-all

	Method	SST-2	MRPC	CoLA	RTE	QNLI	STS-B	MNLI	QQP	Avg.
Best Performance on GLUE Dev										
Full-model Parameter- efficient fine-tuning	Fine-tuning	<u>91.63</u>	<u>90.94</u>	62.08	66.43	89.95	89.76	<u>83.23</u>	87.35	<u>82.67</u>
	Adapter	91.86	89.86	<u>61.51</u>	<u>71.84</u>	90.55	<u>88.63</u>	<u>83.14</u>	<u>86.78</u>	83.02
	Prefix-tuning	90.94	91.29	55.37	76.90	<u>90.39</u>	87.19	81.15	83.30	82.07
	LoRA	91.51	90.03	60.47	71.48	<u>89.93</u>	85.65	82.51	85.98	82.20

Boldface: best performance

Underline: second best performance

Schedule

- 17:00 – 17:10 **Part 1** Introduction [Hung-yi]
- 17:10 – 17:40 **Part 2** Why do PLMs work [Hung-yi]
- 17:40 – 18:20 **Part 3** How to use PLMs: Contrastive Learning for PLMs [Yung-Sung]
- 18:20 – 18:30 Q&A for Part 1+2+3
- 18:30 – 18:40 Break
- 18:40 – 19:05 **Part 4** How to use PLMs: Parameter-efficient fine-tuning [Cheng-Han]
- 19:05 – 19:50 **Part 5** How to use PLMs: Using PLMs with different amounts of data [Cheng-Han]
- 19:50 – 20:00 Conclusion and Future work + Q&A



2022 AACL-IJCNLP

Part 5: How do PLMs work: Using PLMs with different amounts of data

Cheng-Han Chiang

National Taiwan University



Using PLMs with different amounts of data

- Our goal: fine-tune a model for a target downstream task using a PLM
 - Traditionally, we assume that we have sufficient amount of data for the target task



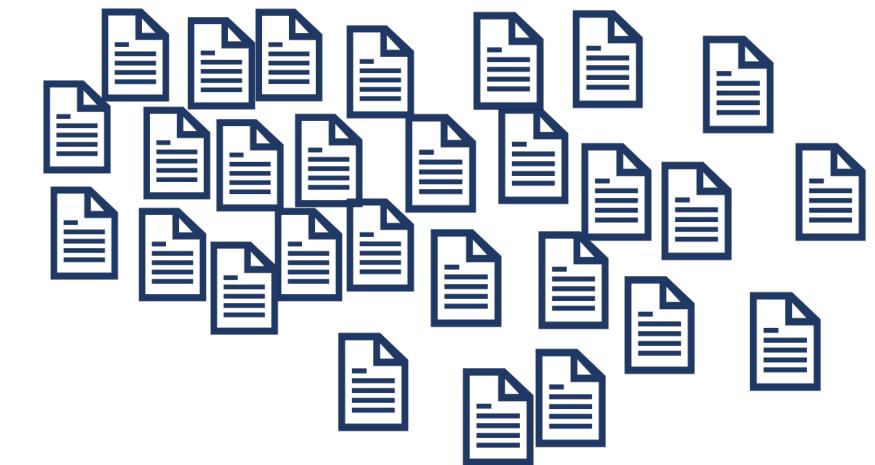
Target task dataset
(labeled)

Using PLMs with different amounts of data

- Our goal: fine-tune a model for a target downstream task using a PLM
 - Sometimes, we have additional labeled dataset for other datasets



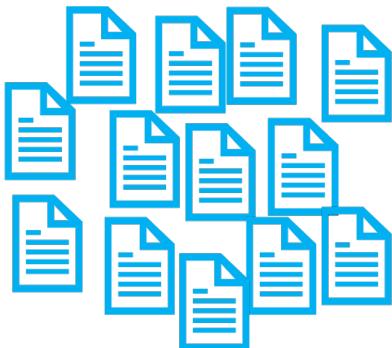
Target task dataset
(labeled)



Datasets of other tasks
(labeled)

Using PLMs with different amounts of data

- Our goal: fine-tune a model for a target downstream task using a PLM
 - Sometimes, labeled data for the target task is scarce



Target task dataset
(labeled)

Using PLMs with different amounts of data

- Our goal: fine-tune a model for a target downstream task using a PLM
 - Sometimes, we only have a few labeled data for the target task, and we have unlabeled dataset related to the target task



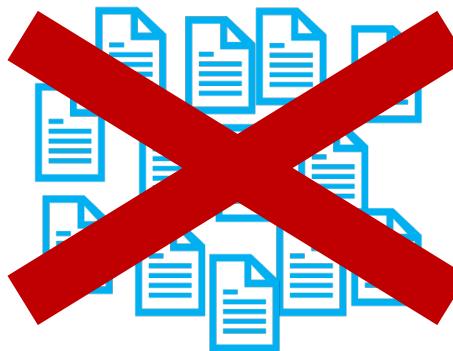
Target task dataset
(labeled)



Target task dataset (Unlabeled)

Using PLMs with different amounts of data

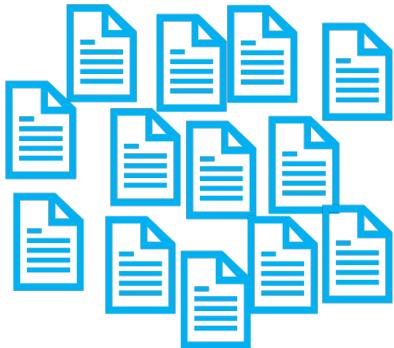
- Our goal: fine-tune a model for a target downstream task using a PLM
 - Sometimes, we have no labeled data for the target task



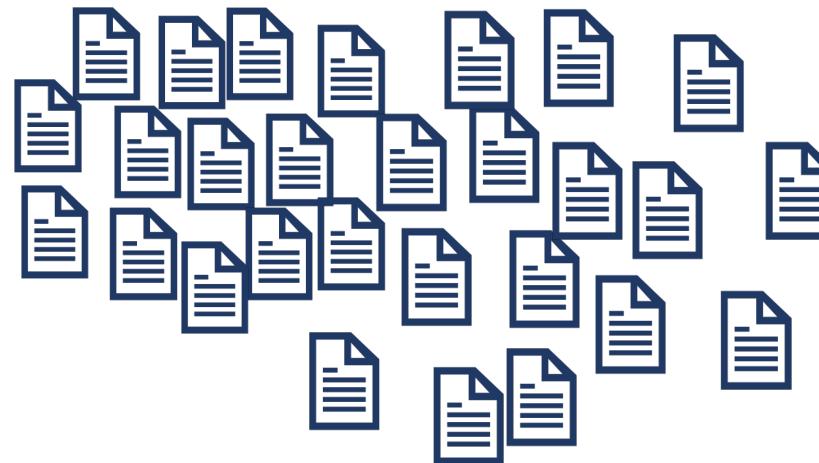
Target task dataset
(labeled)

Using PLMs with different amounts of data

- Our goal: fine-tune a model for a target downstream task using a PLM
 - How to use PLMs with different amount of data?



Target task dataset
(labeled)



Datasets of other tasks
(labeled)



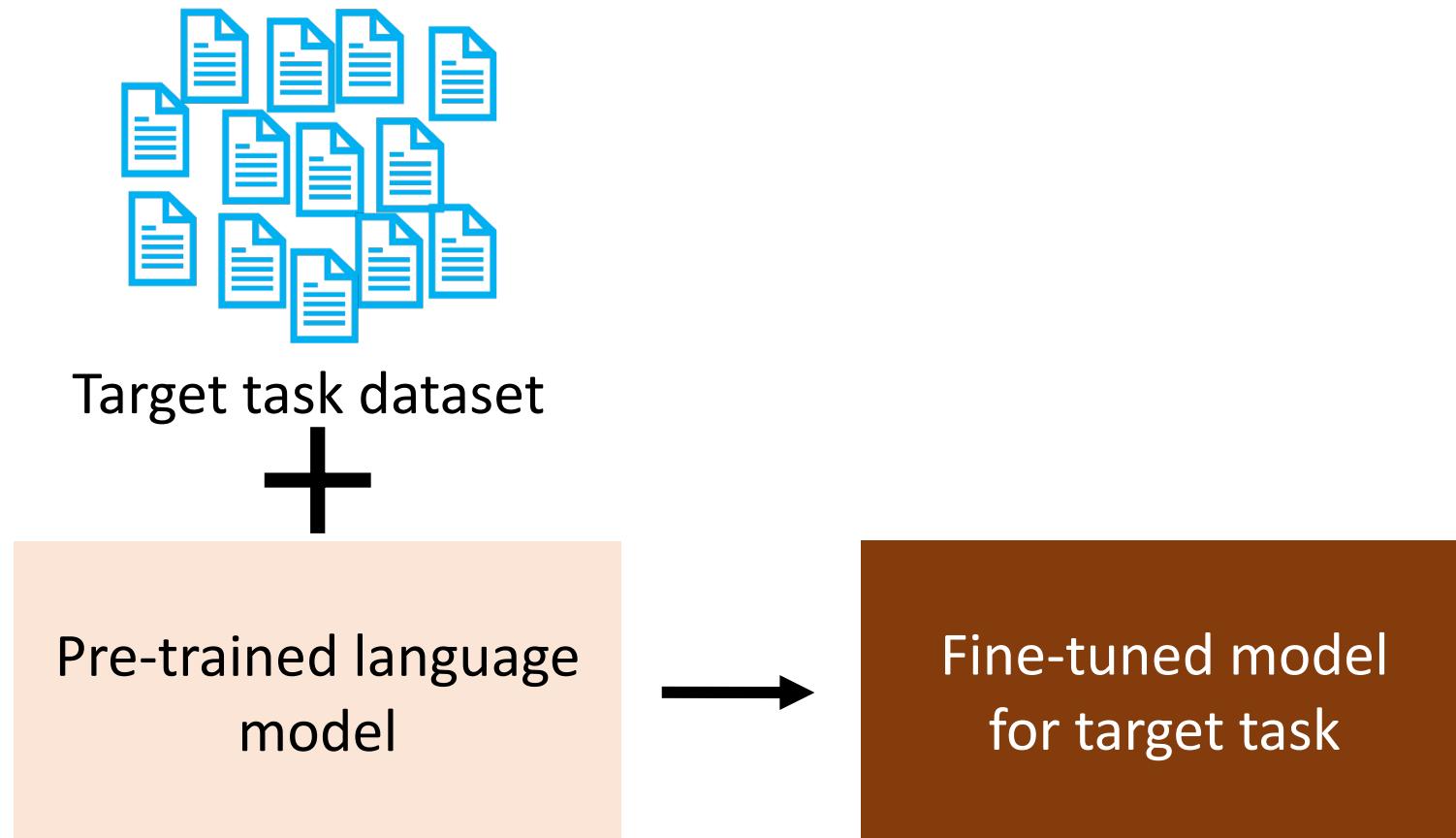
Data related to target task
(Unlabeled)



Part 5:
How do PLMs work:
Using PLMs with different amounts of data
**5-1: Intermediate-task fine-tuning:
using labeled data from other tasks**

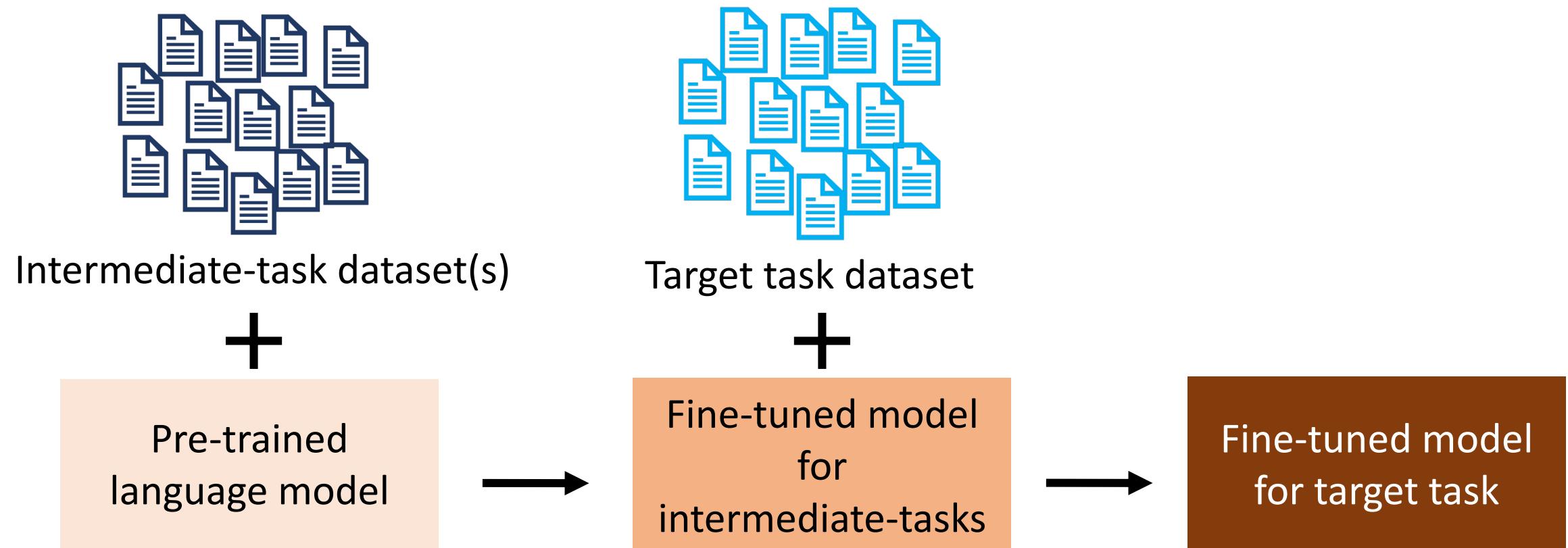
Intermediate-task fine-tuning

- Goal: Obtain a model for task (target task)
 - Standard supervised learning



Intermediate-task fine-tuning

- Goal: Obtain a model for task (target task)
 - Intermediate-task fine-tuning: transfer the knowledge from a model fine-tuned on other tasks (intermediate-tasks)



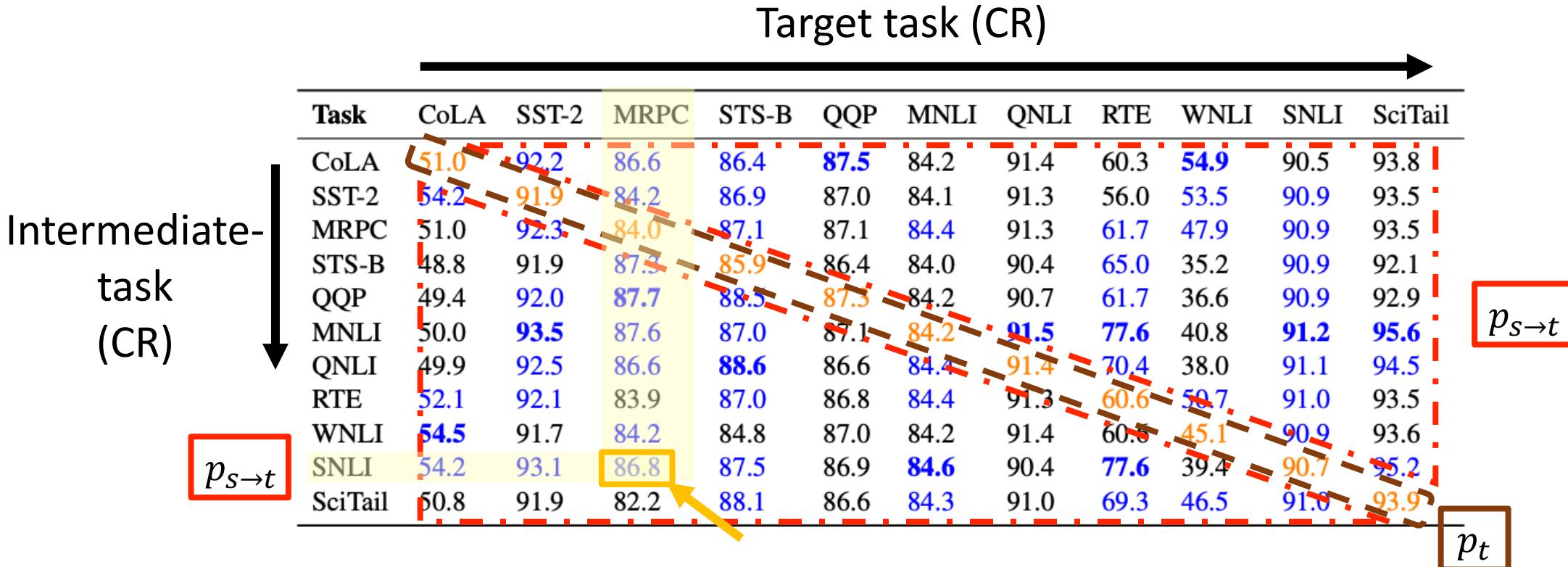
Intermediate-task fine-tuning

- What kind of intermediate tasks can help target task?
 - This paper studies the transferability of 33 datasets, which can be categorized into three types: classification (CR), question answering (QA), and sequence labeling (SL)

Task		
<i>text classification/regression (CR)</i>	<i>question answering (QA)</i>	<i>sequence labeling (SL)</i>
SNLI (Bowman et al., 2015)	SQuAD-2 (Rajpurkar et al., 2018)	ST (Bjerva et al., 2016)
MNLI (Williams et al., 2018)	NewsQA (Trischler et al., 2017)	CCG (Hockenmaier and Steedman, 2007)
QQP (Iyer et al., 2017)	HotpotQA (Yang et al., 2018)	Parent (Liu et al., 2019a)
QNLI (Wang et al., 2019b)	SQuAD-1 (Rajpurkar et al., 2016)	GParent (Liu et al., 2019a)
SST-2 (Socher et al., 2013)	DuoRC-p (Saha et al., 2018)	GGParent (Liu et al., 2019a)
SciTail (Khot et al., 2018)	DuoRC-s (Saha et al., 2018)	POS-PTB (Marcus et al., 1993)
CoLA (Warstadt et al., 2019)	DROP (Dua et al., 2019)	GED (Yannakoudakis et al., 2011)
STS-B (Cer et al., 2017)	WikiHop (Welbl et al., 2018)	NER (Tjong Kim Sang and De Meulder, 2003)
MRPC (Dolan and Brockett, 2005)	BoolQ (Clark et al., 2019)	POS-EWT (Silveira et al., 2014)
RTE (Dagan et al., 2005, et seq.)	ComQA (Abujabal et al., 2019)	Conj (Ficler and Goldberg, 2016)
WNLI (Levesque, 2011)	CQ (Bao et al., 2016)	Chunk (Tjong Kim Sang and Buchholz, 2000)

Intermediate-task fine-tuning

- What kind of intermediate tasks can help target task?
 - p_t : the performance of directly fine-tuning on target task t
 - $p_{s \rightarrow t}$: the performance of transferring from intermediate-task s to a target task t



Intermediate-task fine-tuning

- What kind of intermediate tasks can help target task?
 - p_t : the performance of directly fine-tuning on target task t
 - $p_{s \rightarrow t}$: the performance of transferring from intermediate-task s to a target task t

Target task (CR)

Task	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI	SNLI	SciTail
CoLA	51.0	92.2	86.6	86.4	87.5	84.2	91.4	60.3	54.9	90.5	93.8
SST-2	54.2	91.9	84.2	86.9	87.0	84.1	91.3	56.0	53.5	90.9	93.5
MRPC	51.0	92.3	84.0	87.1	87.1	84.4	91.3	61.7	47.9	90.9	93.5
STS-B	48.8	91.9	87.3	85.9	86.4	84.0	90.4	65.0	35.2	90.9	92.1
QQP	49.4	92.0	87.7	88.5	87.3	84.2	90.7	61.7	36.6	90.9	92.9
MNLI	50.0	93.5	87.6	87.0	87.1	84.2	91.5	77.6	40.8	91.2	95.6
QNLI	49.9	92.5	86.6	88.6	86.6	84.4	91.4	70.4	38.0	91.1	94.5
RTE	52.1	92.1	83.9	87.0	86.8	84.4	91.3	60.6	50.7	91.0	93.5
WNLI	54.5	91.7	84.2	84.8	87.0	84.2	91.4	60.6	45.1	90.9	93.6
SNLI	54.2	93.1	86.8	87.5	86.9	84.6	90.4	77.6	39.4	90.7	95.2
SciTail	50.8	91.9	82.2	88.1	86.6	84.3	91.0	69.3	46.5	91.0	93.9

Intermediate-task fine-tuning

- What kind of intermediate tasks can help target task?
 - The relative transfer gain is defined as $g_{s \rightarrow t} = \frac{p_{s \rightarrow t} - p_t}{p_t}$
 - Same type of tasks is the most beneficial

		FULL → FULL			
		↓src,tgt→	CR	QA	SL
		CR	6.3 (11)	3.4 (10)	0.3 (10)
		QA	3.2 (10)	9.5 (11)	0.3 (9)
		SL	5.3 (8)	2.5 (10)	0.5 (11)

A summary of our transfer results for each combination of the three task classes in the three data regimes. Each cell represents the relative gain of the *best* source task in the source class (row) for a given target task, averaged across all of target tasks in the target class (column). In parentheses, we additionally report the number of target tasks (out of 11) for which at least one source task results in a positive transfer gain. The diagonal cells indicate in-class transfer.

Intermediate-task fine-tuning

- Does the dataset size affect the transferability of intermediate tasks?
 - Limited dataset size: **1K training samples** only
 - Intermediate-task transfer is beneficial even when the intermediate-task or the target task has limited data

Intermediate (src)→target

FULL → LIMITED			
↓src,tgt→	CR	QA	SL
CR	56.9 (11)	36.8 (10)	2.0 (10)
QA	44.3 (11)	63.3 (11)	5.3 (11)
SL	45.6 (11)	39.2 (6)	20.9 (11)

LIMITED → LIMITED			
↓src,tgt→	CR	QA	SL
CR	23.7 (11)	7.3 (11)	1.1 (11)
QA	37.3 (11)	49.3 (11)	4.2 (11)
SL	29.3 (10)	30.0 (8)	10.2 (11)

Intermediate-task fine-tuning

- When fine-tuning the whole model, we will have a full-sized model for each intermediate task

13B

Fine-tuned Model
for Intermediate-Task A

13B

Fine-tuned Model
for Intermediate-Task B

13B

Fine-tuned Model
for Intermediate-Task C

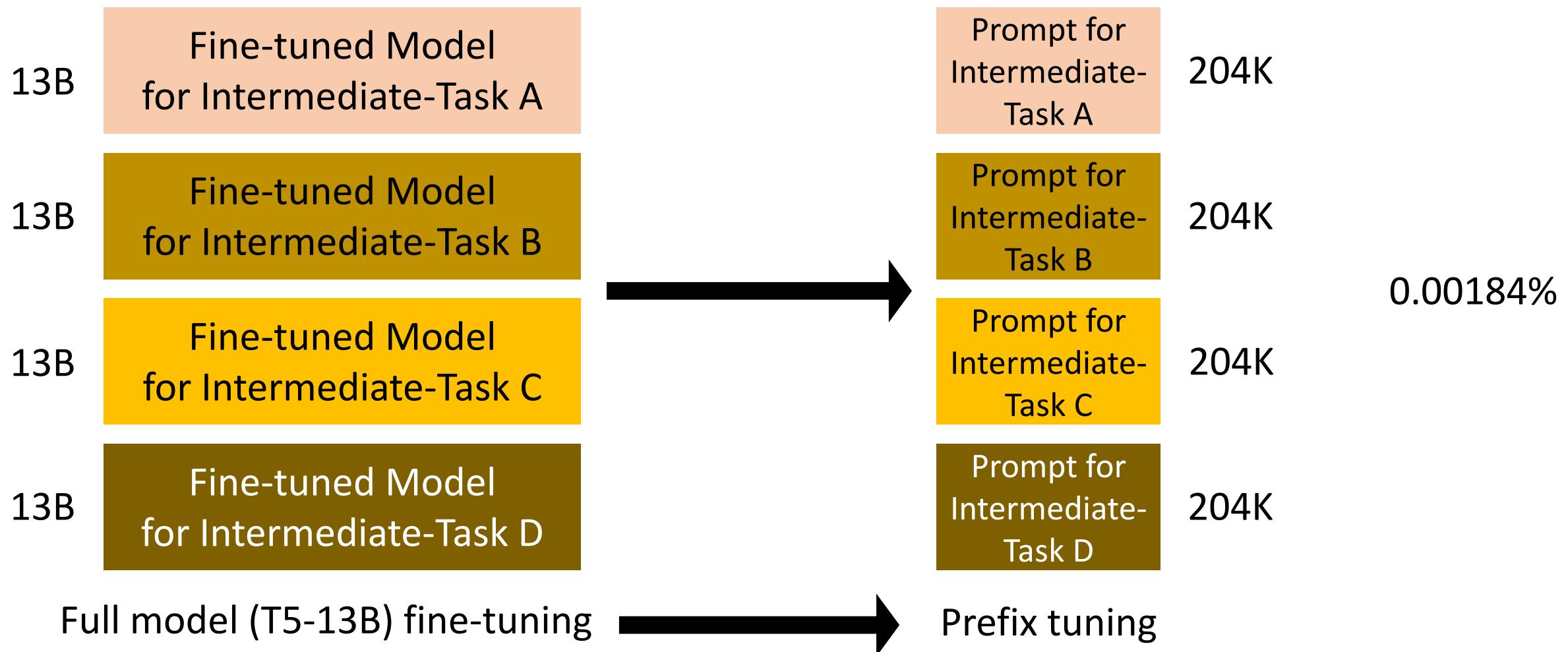
13B

Fine-tuned Model
for Intermediate-Task D

Full model (T5-13B) fine-tuning

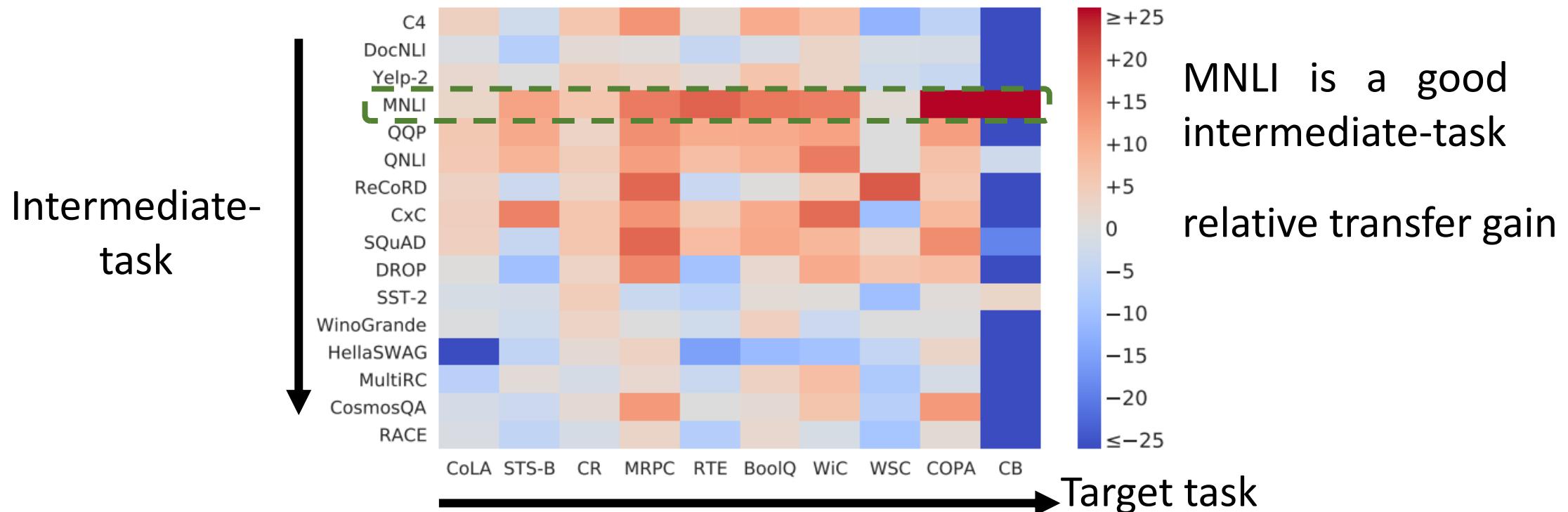
Intermediate-task fine-tuning

- When fine-tuning with soft prompt tuning, we only need to transfer the prompt embedding instead of the whole model



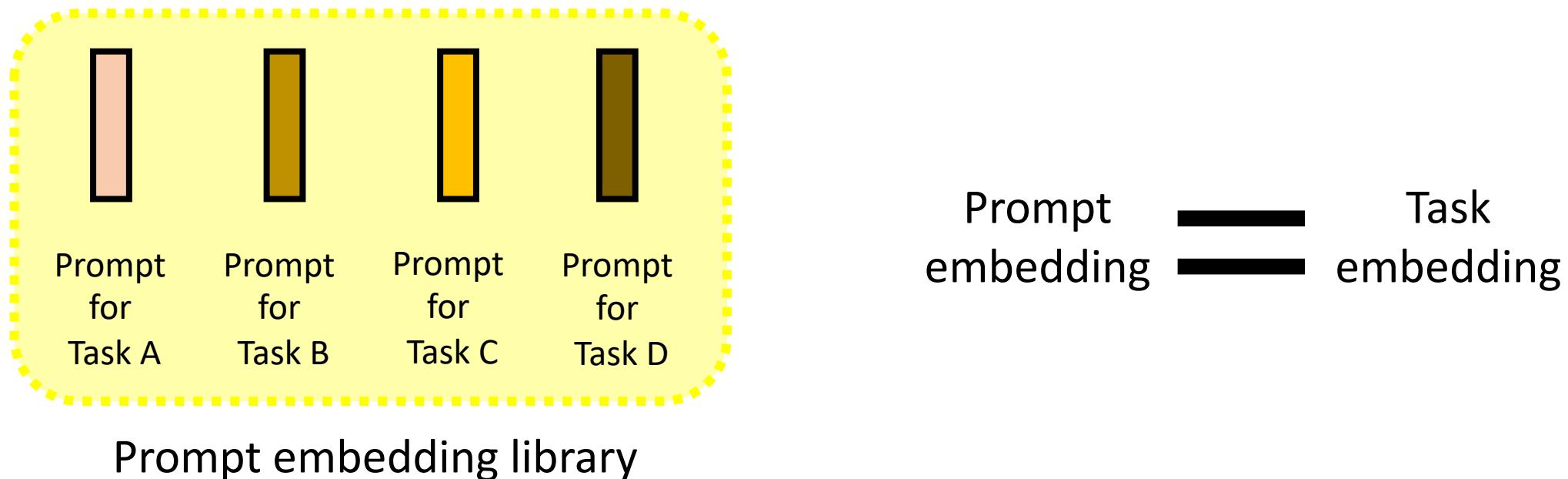
Intermediate-task fine-tuning

- **Soft Prompt Transfer (SPoT):** Using soft prompts for transferring
 - SPoT yields positive transfer in many cases



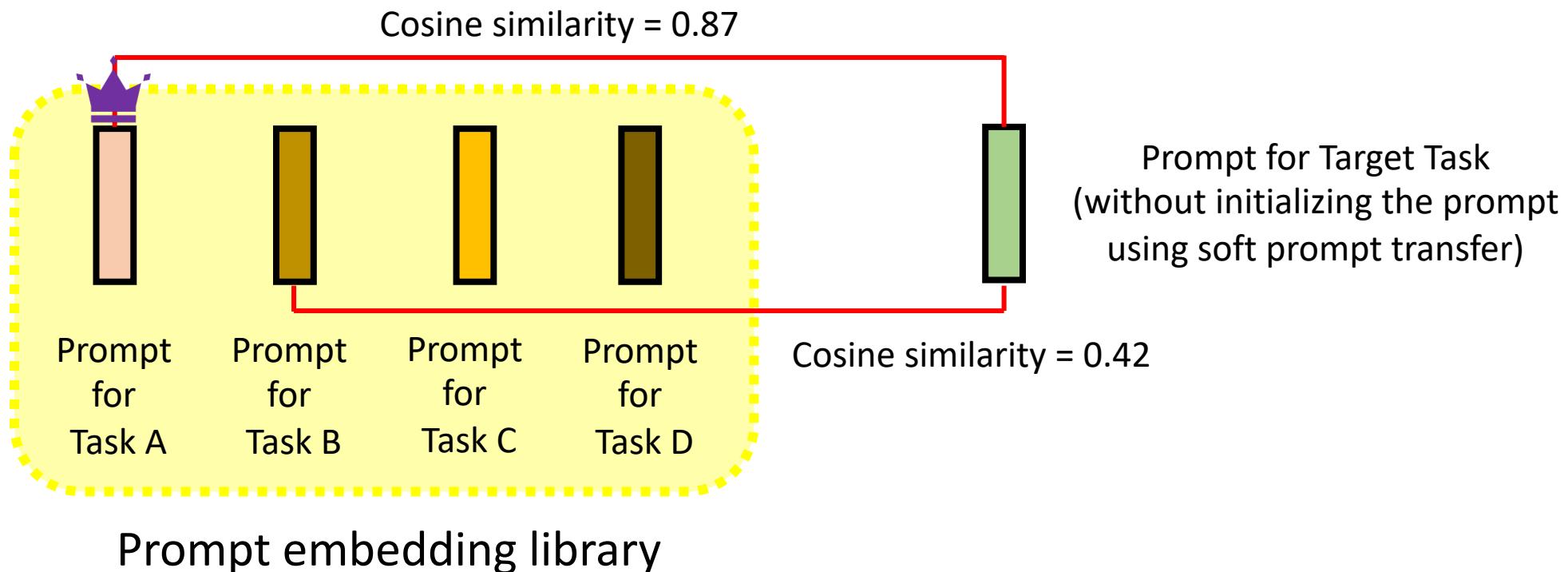
Intermediate-task fine-tuning

- **Soft Prompt Transfer (SPoT):** The soft prompt of a task can be used as the task embedding of that task.



Intermediate-task fine-tuning

- **Soft Prompt Transfer (SPoT):** Given a novel task, we can first train only using the novel task, and find a intermediate task whose task embedding is most similar to the novel task and use it to transfer

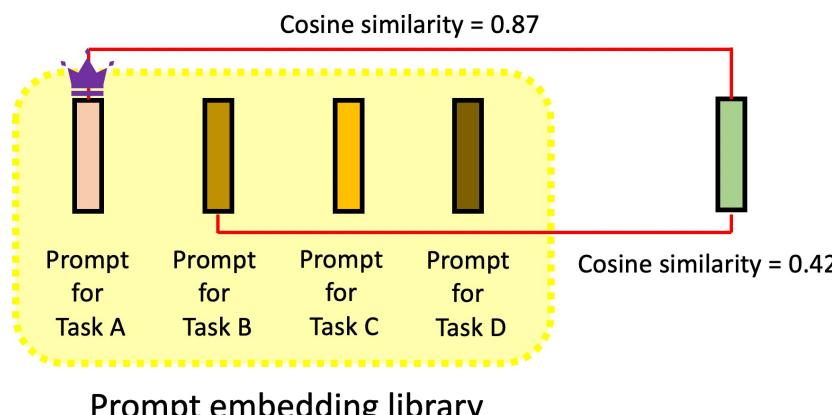


Intermediate-task fine-tuning

- **Soft Prompt Transfer (SPoT):** Selecting the best intermediate-task soft prompt

w/o SPoT:

Oracle SPoT:



Method	Change		Avg. score (16 target tasks)	
	Abs.	Rel.		
BASELINE	-	-	74.7	_{0.7}
BRUTE-FORCE SEARCH ($k = 48$)				
ORACLE	6.0 _{0.5}	26.5 _{1.1}	80.7	_{0.0}
COSINE SIMILARITY BEST OF TOP- k				
$k = 1$	1.5 _{0.5}	11.7 _{1.1}	76.2	_{0.1}
$k = 3$	2.7 _{0.6}	16.6 _{1.1}	77.4	_{0.3}
$k = 6$	3.8 _{0.1}	20.0 _{1.1}	78.5	_{0.5}
$k = 9$	4.5 _{0.4}	22.2 _{1.1}	79.2	_{0.1}
$k = 12$	5.0 _{0.9}	23.6 _{2.2}	79.7	_{0.4}
$k = 15$	5.4 _{0.8}	24.9 _{1.8}	80.1	_{0.3}



Part 5:
How do PLMs work:
Using PLMs with different amounts of data
**5-1.1: Multi-task fine-tuning:
using labeled data from other tasks**

Multi-task fine-tuning

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



How to weight the loss of different tasks?



Part 5:

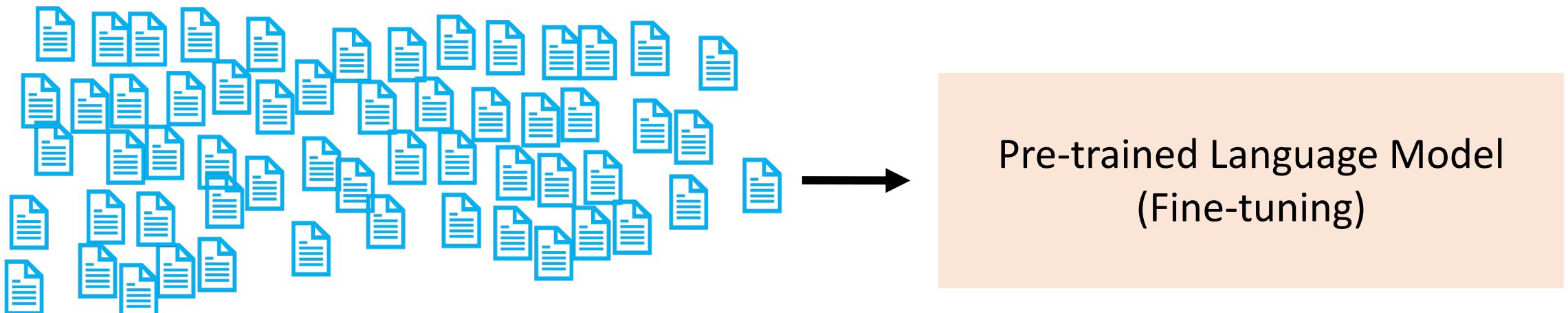
How do PLMs work:

Using PLMs with different amounts of data

5-2: Prompt tuning for few-shot learning

Prompt tuning for few-shot learning

- Standard fine-tuning mostly assumes a large amount of labeled data

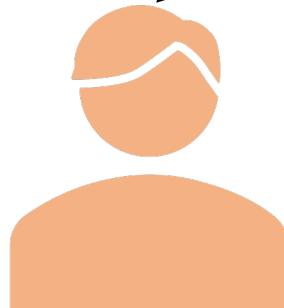


MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE
392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k

(size of training set)

- [CLS] Jack likes dog. [SEP] Jack loves ice cream. [SEP] >>> **1**
- [CLS] The spring break is coming soon. [SEP] The spring break was over. [SEP] >>> **2**
- [CLS] I am going to have dinner. [SEP] I am going to eat something. [SEP] >>> **0**
-
- [CLS] Mary likes pie. [SEP] Mary hates pie. [SEP] >>> ?

0: entailment
1: neutral
2: contradiction

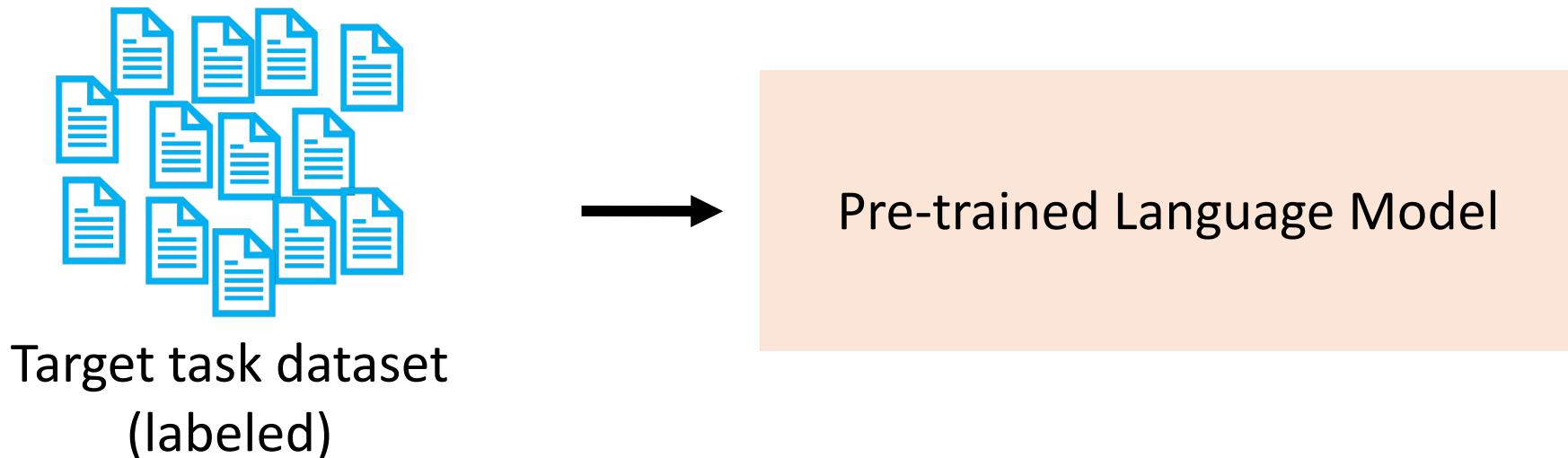


NLI model

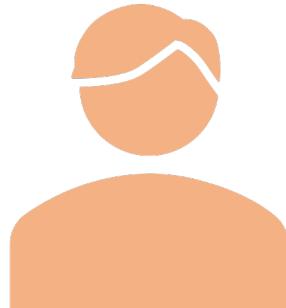
Natural language inference (NLI): premise + hypothesis

Prompt tuning for few-shot learning

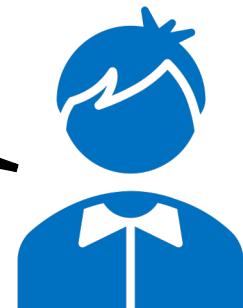
- **Data scarcity** in downstream tasks is very common
- Few-shot learning: We have some labeled training data
 - "Some" \approx less than a hundred



- [CLS] The spring break is coming soon. [SEP] The spring break was over. [SEP] >>> 2
- [CLS] I am going to have dinner. [SEP] I am going to eat something. [SEP] >>> 0
- [CLS] Mary likes pie. [SEP] Mary hates pie. [SEP] >>> ?



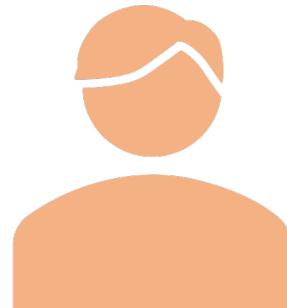
?????



Natural language inference (NLI): premise + hypothesis

Prompt tuning for few-shot learning

- [CLS] The spring break is coming soon. Is it true that the spring break was over? >>> no
- [CLS] I am going to have dinner. Is it true that I am going to eat something? >>> yes
- [CLS] Mary likes pie. Is it true that Mary hates pie. [SEP]
>>> ?



no



Natural language inference (NLI): premise + hypothesis

Prompt tuning for few-shot learning

- By converting the data points in the dataset into **natural language prompts**, the model may be easier to know what it should do

- [CLS] The spring break is coming soon.
[SEP] The spring break was over. [SEP] >>>
contradiction
- [CLS] I am going to have dinner. [SEP] I am
going to eat something. [SEP] >>>
entailment
- [CLS] Mary likes pie. [SEP] Mary hates pie.
[SEP] >>> ?

- [CLS] The spring break is coming soon.
Is it true that the spring break was
over? >>> **no**
- [CLS] I am going to have dinner. **Is it**
true that I am going to eat something?
>>> **yes**
- [CLS] Mary likes pie. **Is it true that**
Mary hates pie. [SEP] >>> ?

Prompt tuning for few-shot learning

- Format the downstream task as a language modelling task with pre-defined templates into natural language **prompts**

verb (used with object)

- 5 to move or **induce to action:**

What prompted you to say that?

- 6 to occasion or **incite; inspire:**

What prompted his resignation?

noun

- 11 the act of prompting.

Prompt tuning for few-shot learning

- What you need in prompt tuning

1. A prompt template
2. A PLM
3. A verbalizer

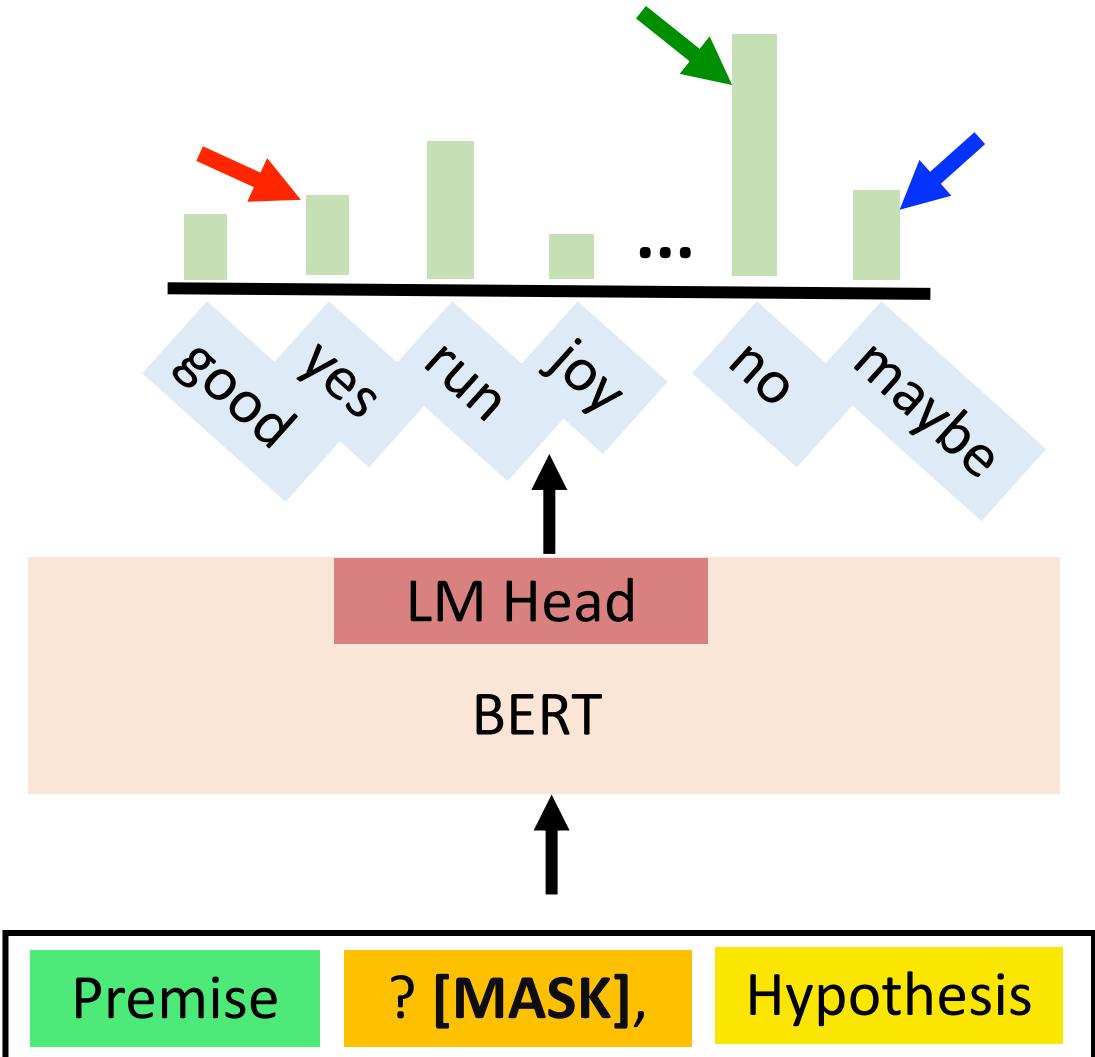
Premise	Mary likes pie.
Hypothesis	Mary hates pie.

Label 2

```
▼ "label" : [  
    0 : "entailment"  
    1 : "neutral"  
    2 : "contradiction"  
]
```

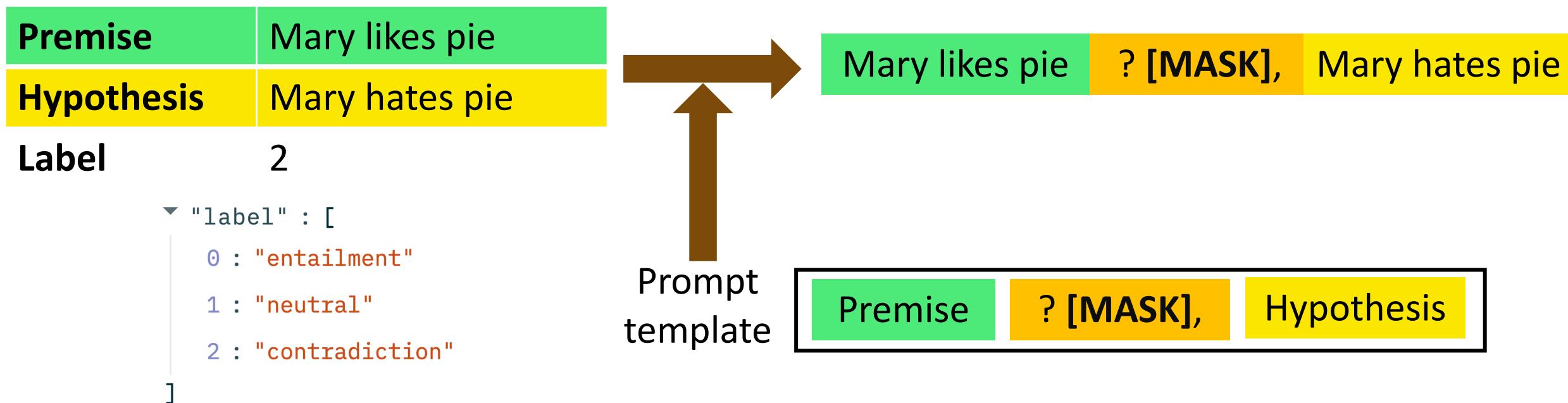
→ {
 yes
 maybe
 no
}

Prompt
template:



Prompt tuning for few-shot learning

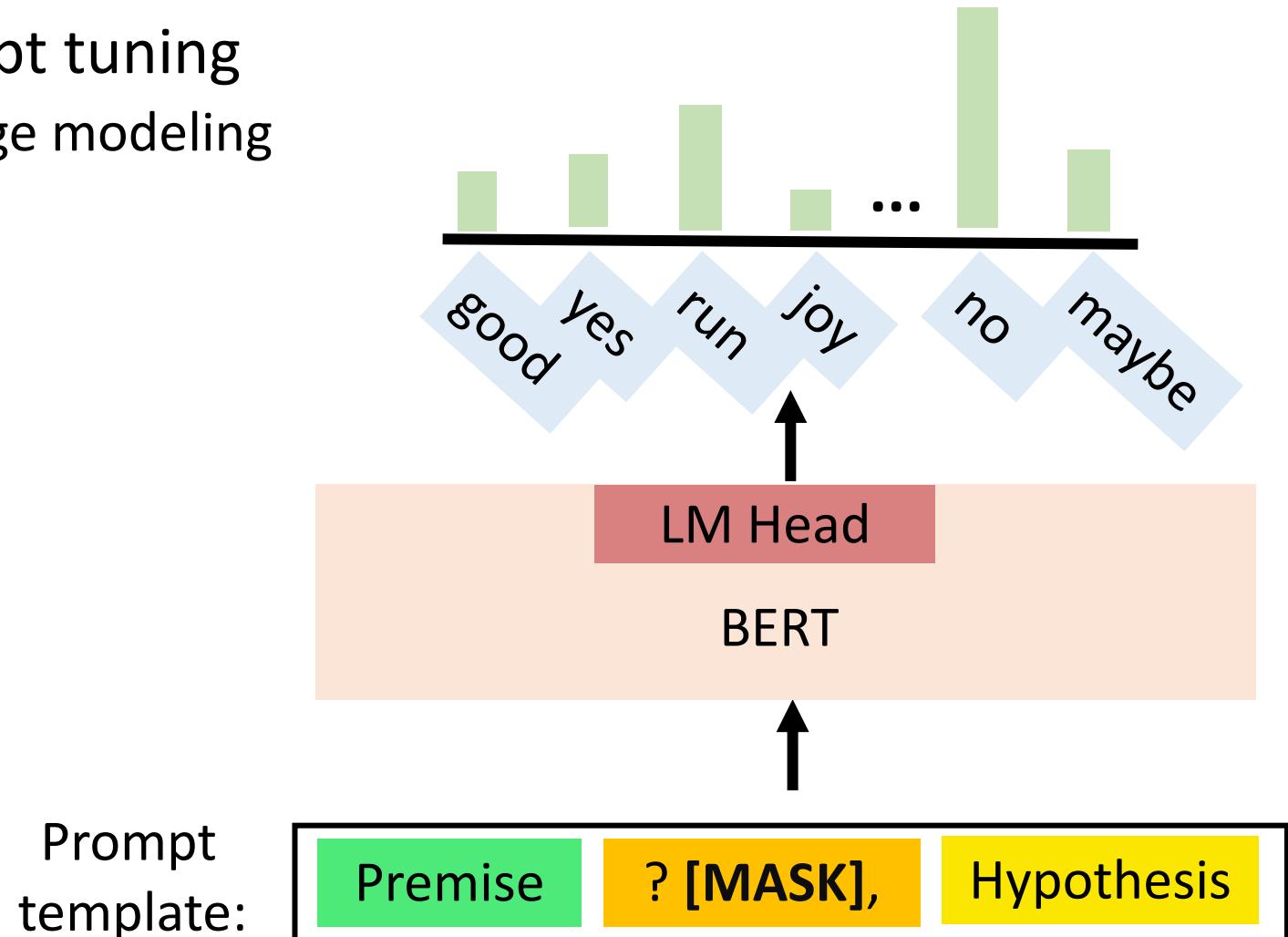
- What you need in prompt tuning
 1. A prompt template: convert data points into a natural language prompt



Schick, Timo, and Hinrich Schütze. "Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference." *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021.

Prompt tuning for few-shot learning

- What you need in prompt tuning
 - 2. A PLM: perform language modeling

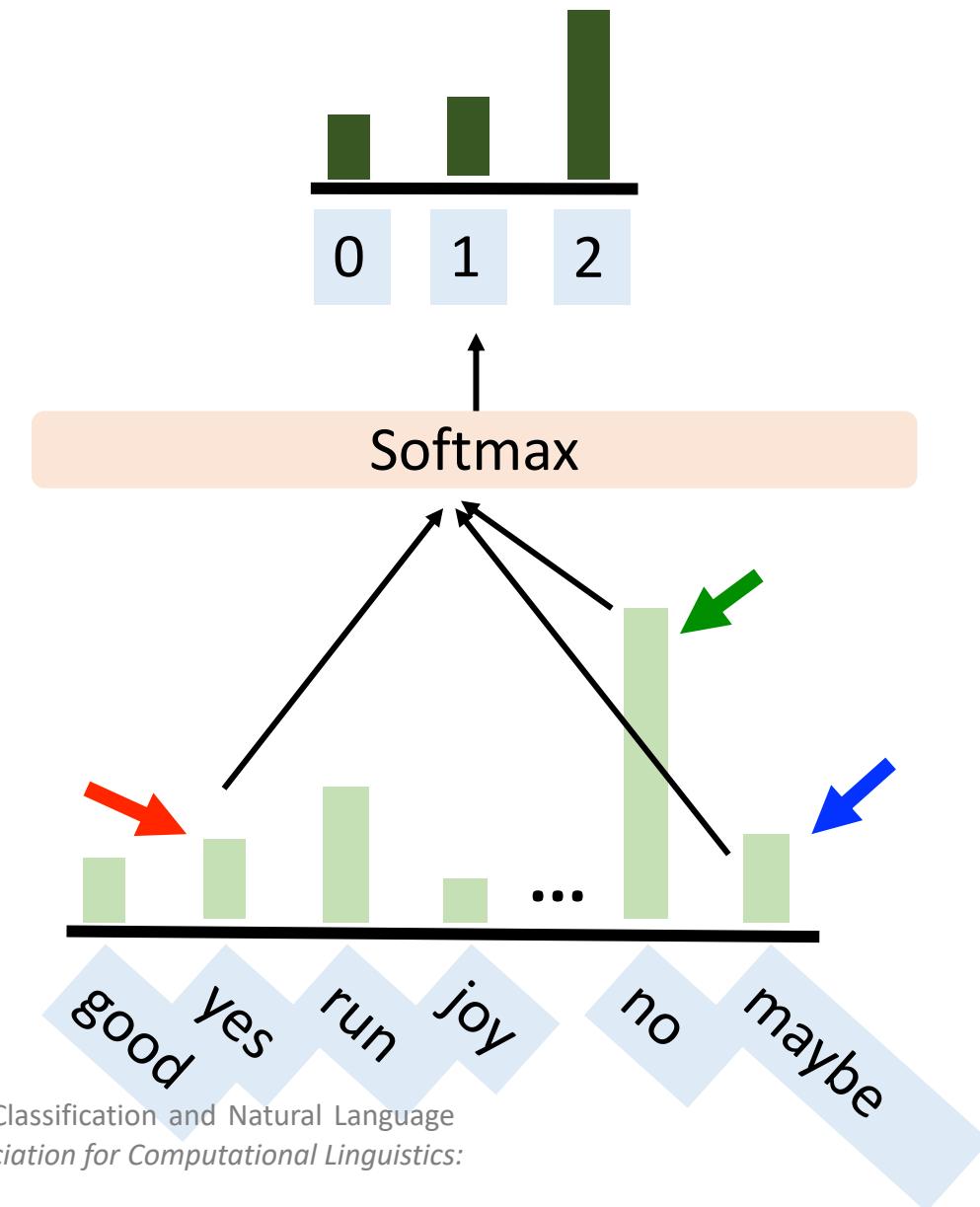
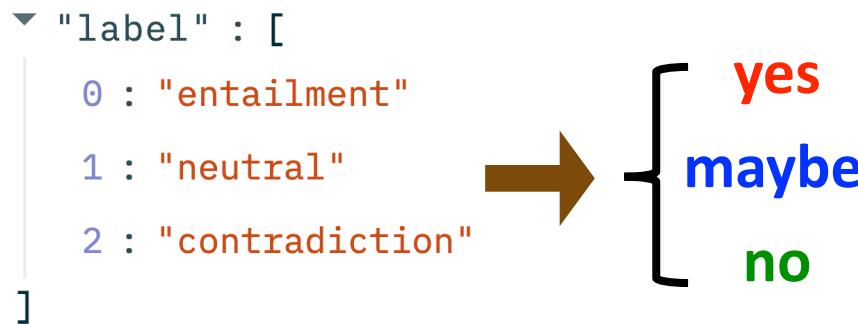


Prompt tuning for few-shot learning

- What you need in prompt tuning

3. A verbalizer: A mapping between the label and the vocabulary

- Which vocabulary should represents the class “entailment”



Prompt tuning for few-shot learning

- Prompt tuning
 - The whole PLM will be fine-tuned

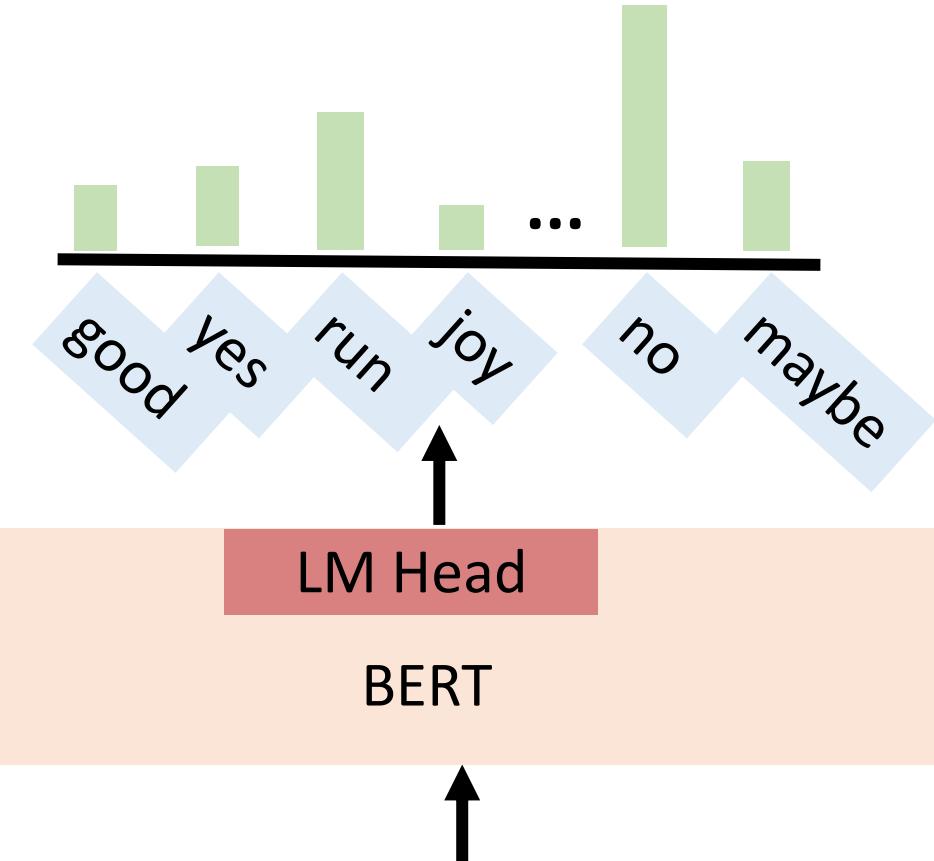
Premise	Mary likes pie.
Hypothesis	Mary hates pie.

Label 2

```
▼ "label" : [  
    0 : "entailment"  
    1 : "neutral"  
    2 : "contradiction"  
]
```

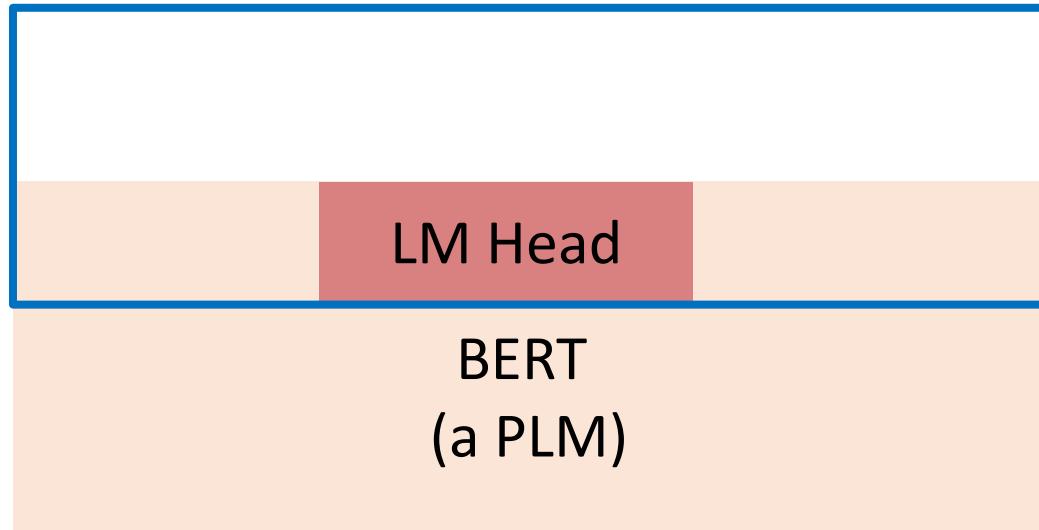
Prompt
template:

Premise	? [MASK],	Hypothesis
---------	-----------	------------

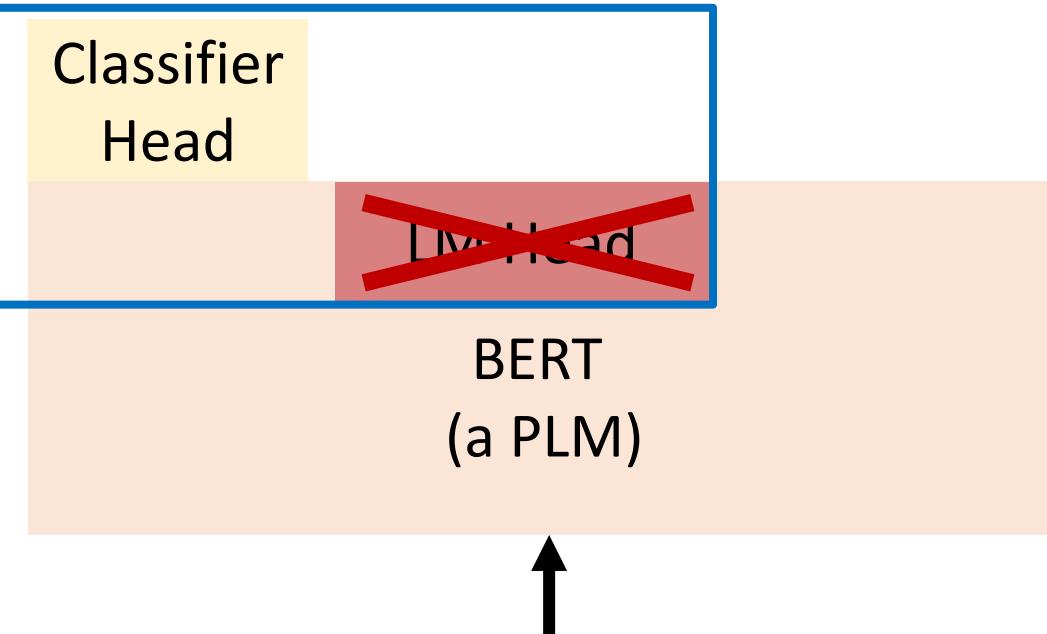


Prompt tuning for few-shot learning

- Prompt tuning



- Standard fine-tuning

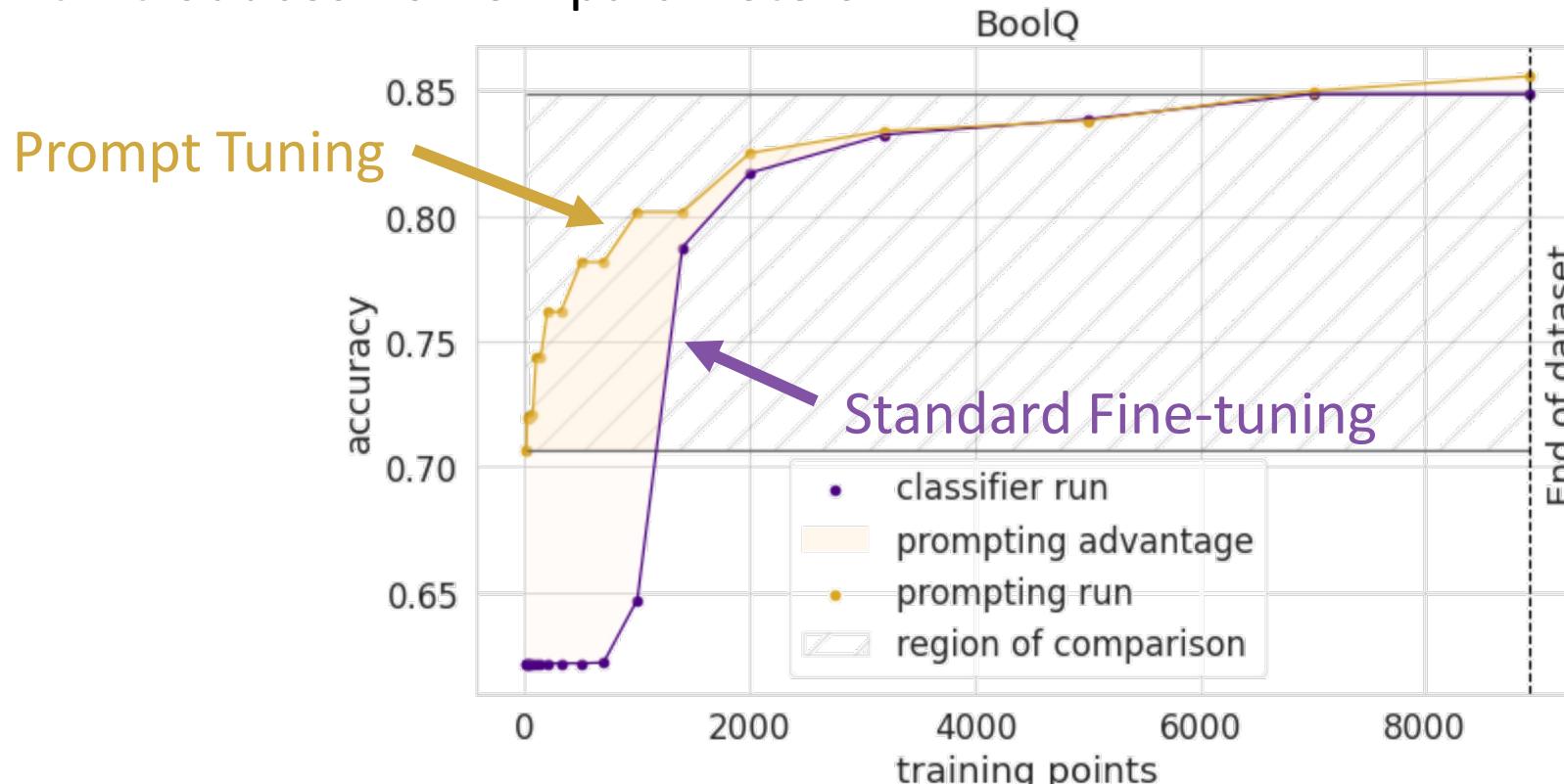


* I omit the [CLS] at the beginning and the [SEP] at the end

Schick, Timo, and Hinrich Schütze. "Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference." *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021.

Prompt tuning for few-shot learning

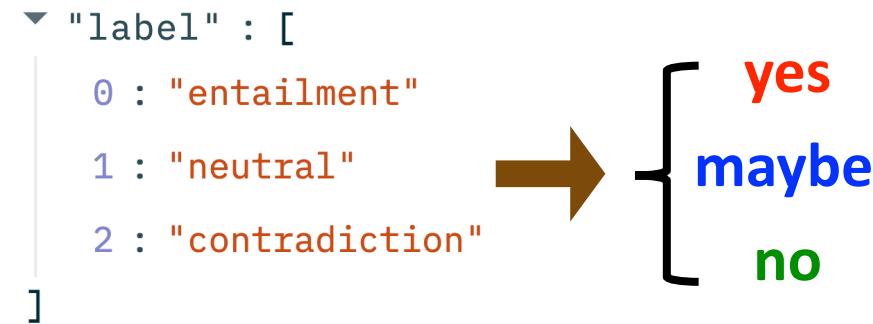
- Prompt tuning has better performance under data scarcity because
 - It incorporates human knowledge
 - It introduces no new parameters



Le Scao, Teven, and Alexander M. Rush. "How many data points is a prompt worth?." *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021.

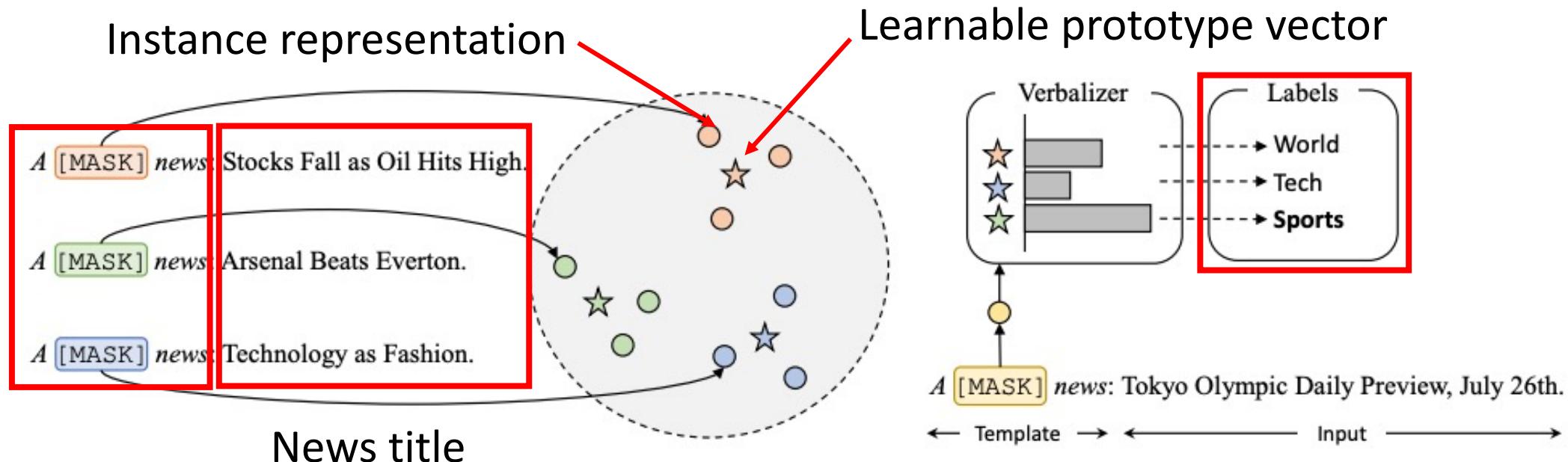
Prompt tuning for few-shot learning

- How to select the verbalizer?
 - 1. Manual design: require task-specific knowledge



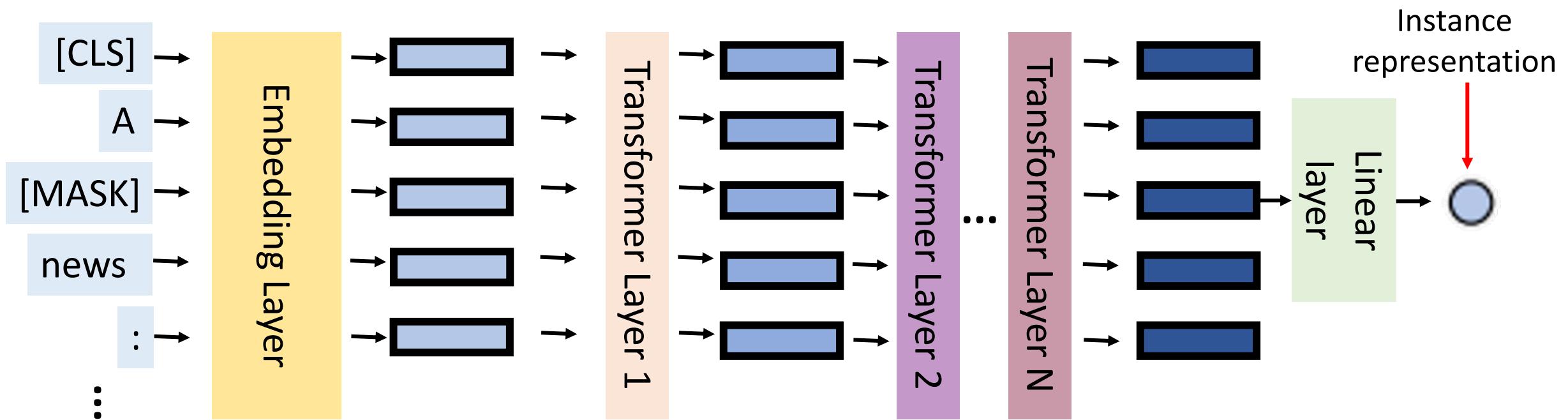
Prompt tuning for few-shot learning

- How to select the verbalizer?
 - 2. Prototypical verbalizer: use learnable prototype vectors to represent a class, instead of using the words in the vocabulary



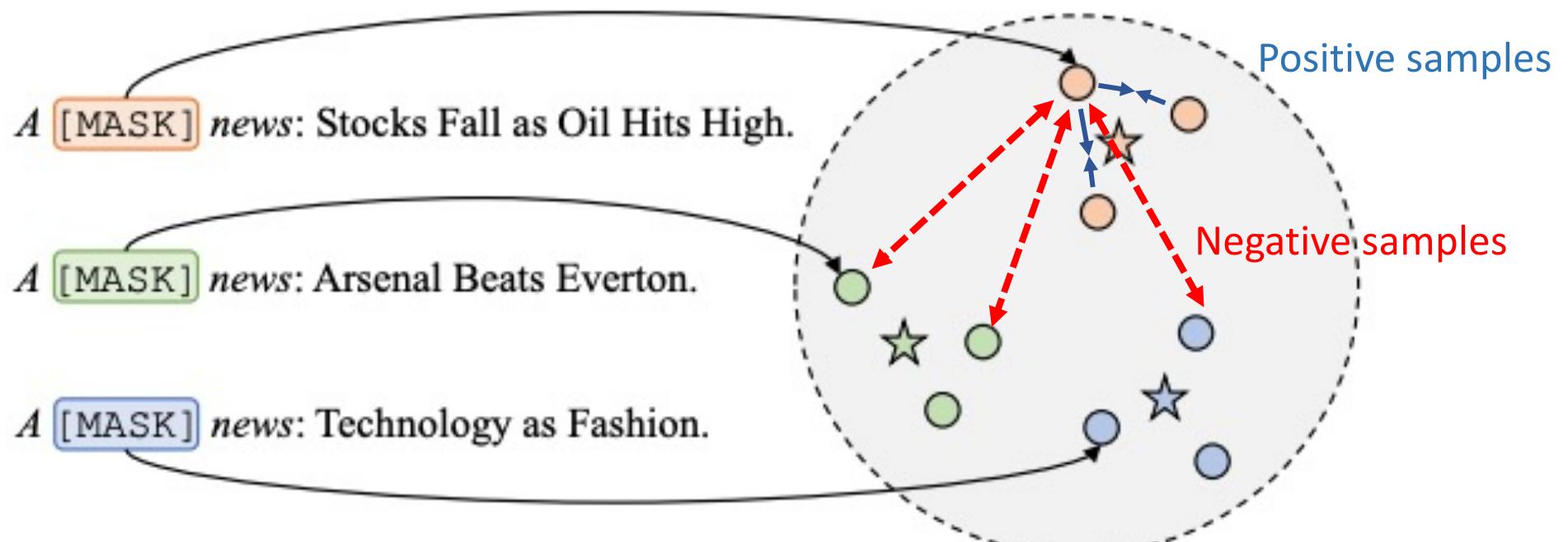
Prompt tuning for few-shot learning

- How to select the verbalizer?
 - 2. Prototypical verbalizer



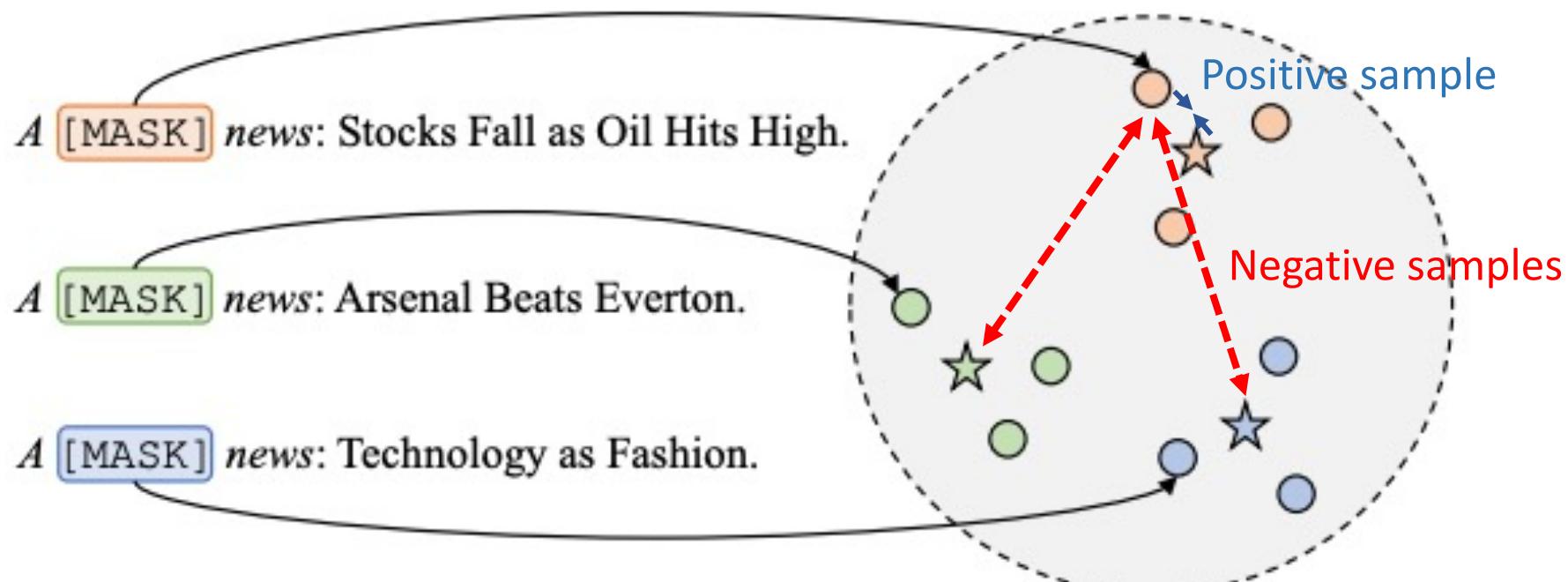
Prompt tuning for few-shot learning

- How to select the verbalizer?
 - 2. Prototypical verbalizer
 - Trained by contrastive learning: (1) instance-instance contrastive



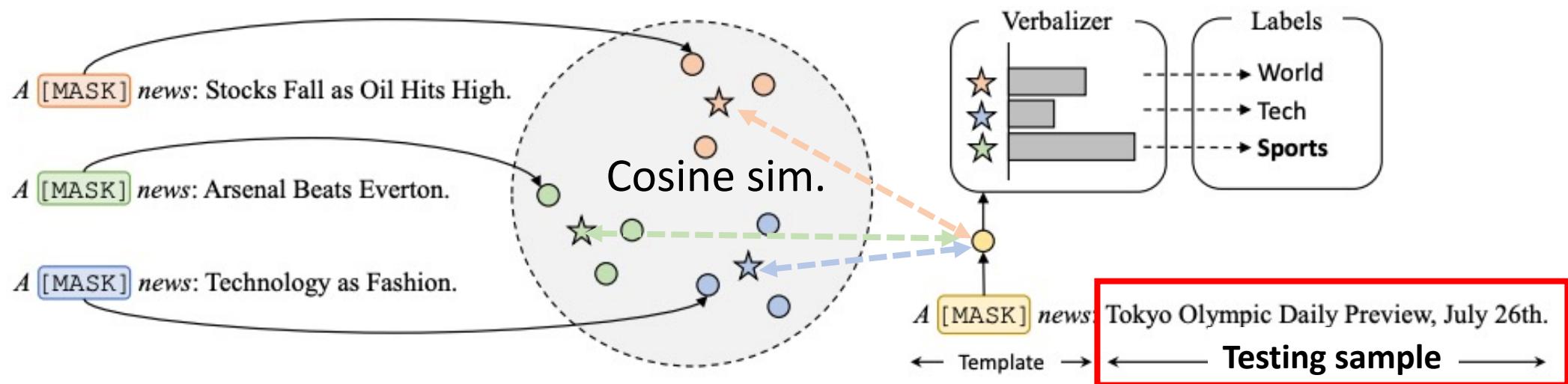
Prompt tuning for few-shot learning

- How to select the verbalizer?
 - 2. Prototypical verbalizer
 - Trained by contrastive learning: (2) instance-prototype contrastive



Prompt tuning for few-shot learning

- How to select the verbalizer?
 - 2. Prototypical verbalizer
 - Inference by finding the prototype that is most similar with the testing data's instance representation



Prompt tuning for few-shot learning

- How to select the verbalizer?

K : Number of training data
for each class



K	Method	AG	DB	Yahoo	Few
0	ManualVerb	75.13	67.06	43.11	20.00
	ManualVerb	76.67	85.47	50.22	41.68
	SearchVerb	41.50	60.06	27.39	20.88
1	ProtoVerb	64.19	72.85	36.12	25.00
	ManualVerb	81.06	93.61	58.65	46.44
	SearchVerb	65.82	78.21	40.71	31.28
2	ProtoVerb	77.34	85.49	46.30	35.72
	ManualVerb	84.73	95.83	61.41	52.54
	SearchVerb	77.43	86.40	51.58	43.10
4	ProtoVerb	81.65	90.91	55.08	48.28
	ManualVerb	85.85	96.46	64.12	56.59
	SearchVerb	82.17	88.41	58.64	50.78
8	ProtoVerb	84.03	95.75	61.40	56.06
	ManualVerb	84.74	96.05	58.77	61.17
	SearchVerb	83.40	92.00	59.66	55.49
16	ProtoVerb	84.48	96.30	64.35	61.29

Manual verbalizer is good most of the time, but it requires task-specific knowledge

Prompt tuning for few-shot learning

- How to select the verbalizer?

K : Number of training data
for each class

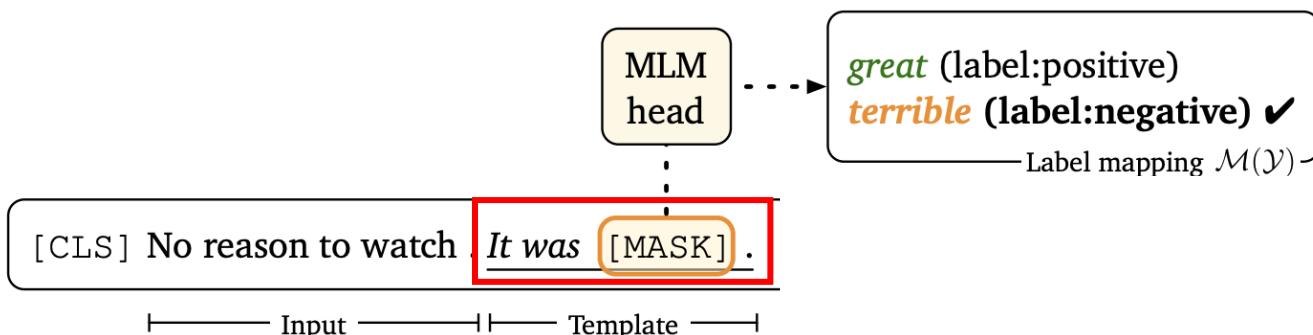


K	Method	AG	DB	Yahoo	Few
0	ManualVerb	75.13	67.06	43.11	20.00
	ManualVerb	76.67	85.47	50.22	41.68
	SearchVerb	41.50	60.06	27.39	20.88
1	ProtoVerb	64.19	72.85	36.12	25.00
	ManualVerb	81.06	93.61	58.65	46.44
	SearchVerb	65.82	78.21	40.71	31.28
2	ProtoVerb	77.34	85.49	46.30	35.72
	ManualVerb	84.73	95.83	61.41	52.54
	SearchVerb	77.43	86.40	51.58	43.10
4	ProtoVerb	81.65	90.91	55.08	48.28
	ManualVerb	85.85	96.46	64.12	56.59
	SearchVerb	82.17	88.41	58.64	50.78
8	ProtoVerb	84.03	95.75	61.40	56.06
	ManualVerb	84.74	96.05	58.77	61.17
	SearchVerb	83.40	92.00	59.66	55.49
16	ProtoVerb	84.48	96.30	64.35	61.29

Prototypical verbalizer requires no task-specific knowledge and can work well even when there is only one label for each class

Prompt tuning for few-shot learning

- Can we further improve the few-shot performance of PLMs?
- LM-BFF: better few-shot fine-tuning of lmodels
 - Core concept: **prompt** + **demonstration**



Prompt tuning for few-shot learning

- LM-BFF
 - Demonstrations can improve the performance of prompt tuning and makes the variance smaller

	MNLI (acc)	MNLI-mm (acc)	SNLI (acc)	QNLI (acc)	RTE (acc)	MRPC (F1)	QQP (F1)	STS-B (Pear.)	
K = 16	Standard fine-tuning	45.8 (6.4)	47.8 (6.8)	48.4 (4.8)	60.2 (6.5)	54.4 (3.9)	76.6 (2.5)	60.7 (4.3)	53.5 (8.5)
	Prompt tuning	68.3 (2.3)	70.5 (1.9)	77.2 (3.7)	64.5 (4.2)	69.1 (3.6)	74.5 (5.3)	65.5 (5.3)	71.0 (7.0)
	+ demonstration (LM-BFF)	70.7 (1.3)	72.0 (1.2)	79.7 (1.5)	69.2 (1.9)	68.7 (2.3)	77.8 (2.0)	69.8 (1.8)	73.5 (5.1)
	Fine-tuning (full) [†]	89.8	89.5	92.6	93.3	80.9	91.4	81.7	91.9

Prompt tuning for few-shot learning

- Question: What's the difference between prompting and probing
- Answer:
 - The concept of “prompting” is first used in recent NLP community for probing the factual knowledge of a PLM

Prompts →	Query	Answer
	Francesco Bartolomeo Conti was born in ____.	Florence
	Adolphe Adam died in ____.	Paris
	English bulldog is a subclass of ____.	dog
	The official language of Mauritius is ____.	English
	Patrick Oboya plays in ____ position.	midfielder
	Hamburg Airport is named after ____.	Hamburg

Prompt tuning for few-shot learning

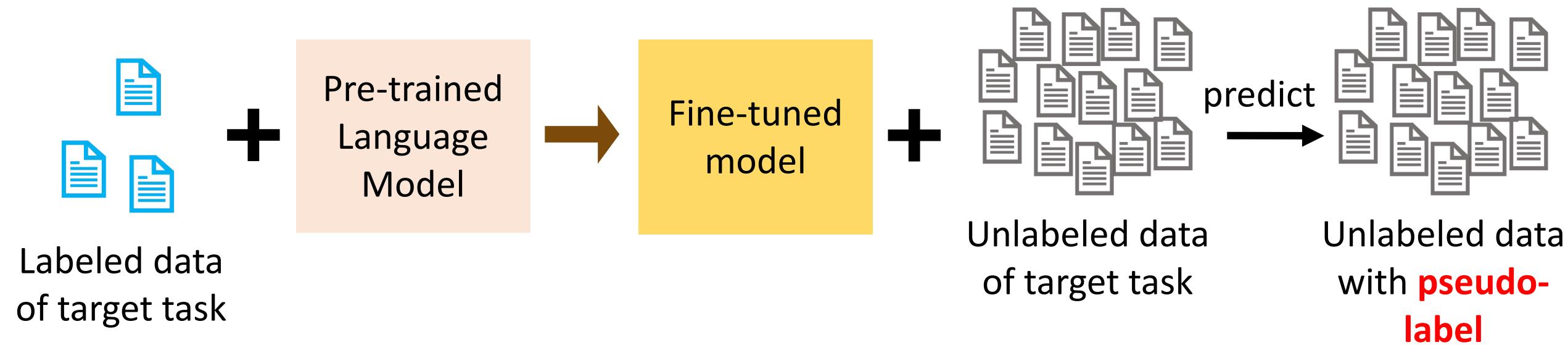
- Question: What's the difference between prompting and probing
- Answer:
 - Probing is the process of exploring what knowledge is encoded in the PLM. PLMs are often fixed during prompting.
 - Prompting means using natural language to query the PLM, perhaps for the downstream task. PLM can be fine-tuned during prompting.
 - The purpose of prompting and probing are different.



Part 5:
How do PLMs work:
Using PLMs with different amounts of data
5-3: Semi-supervised learning with PLMs

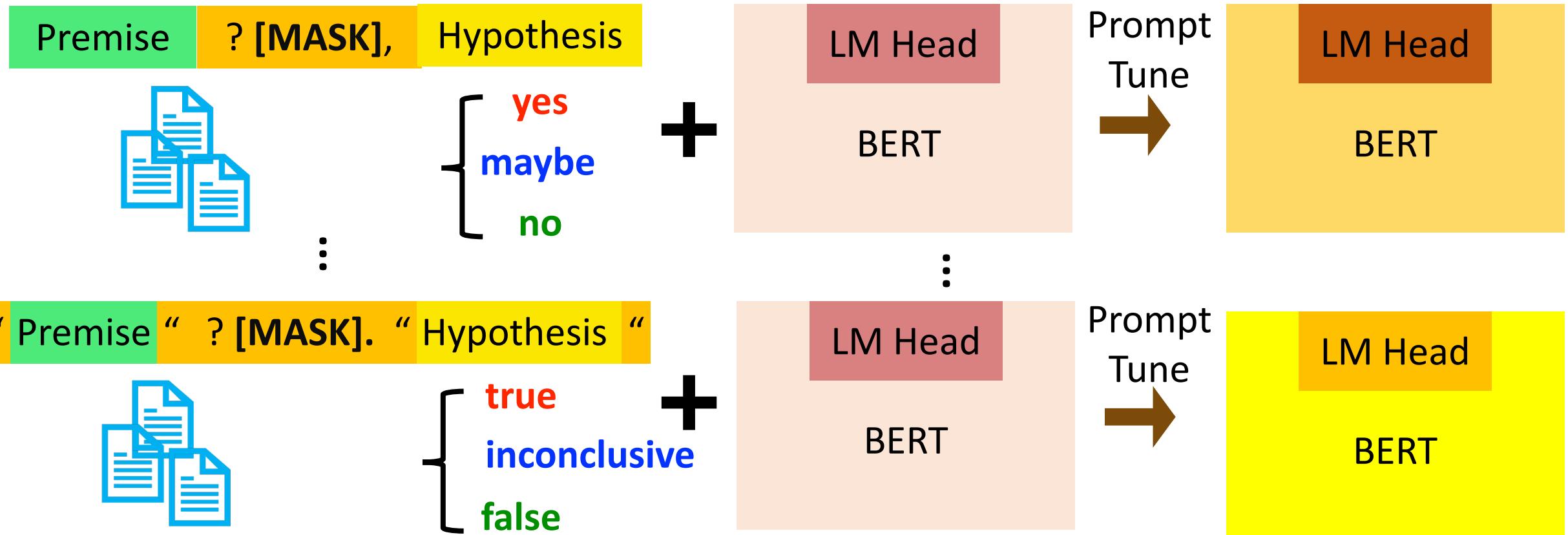
Semi-supervised learning with PLMs

- Semi-Supervised learning: We have some labeled training data and a large amount of unlabeled data
- Core idea: use the labeled data to train a good model and use that model to label the unlabeled data



Semi-supervised learning with PLMs: PET

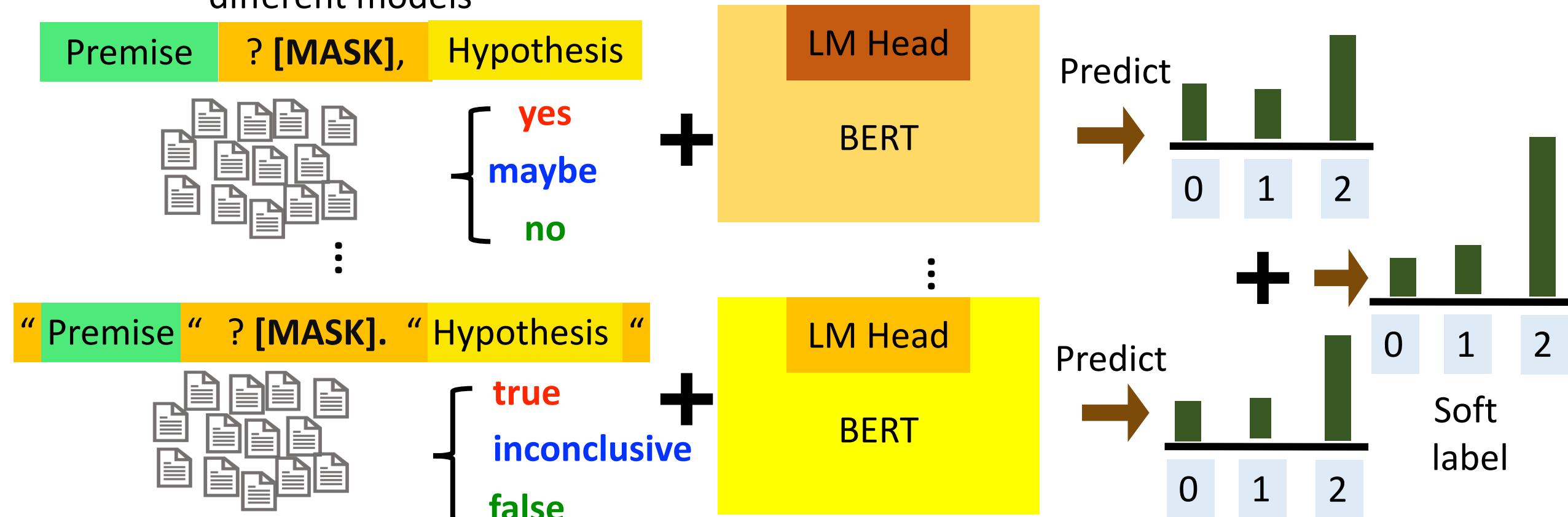
- Pattern-Exploiting Training (PET)
 - Step 1: Use different prompts and verbalizer to prompt-tune different PLMs on the labeled dataset



Schick, Timo, and Hinrich Schütze. "Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference." *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021.

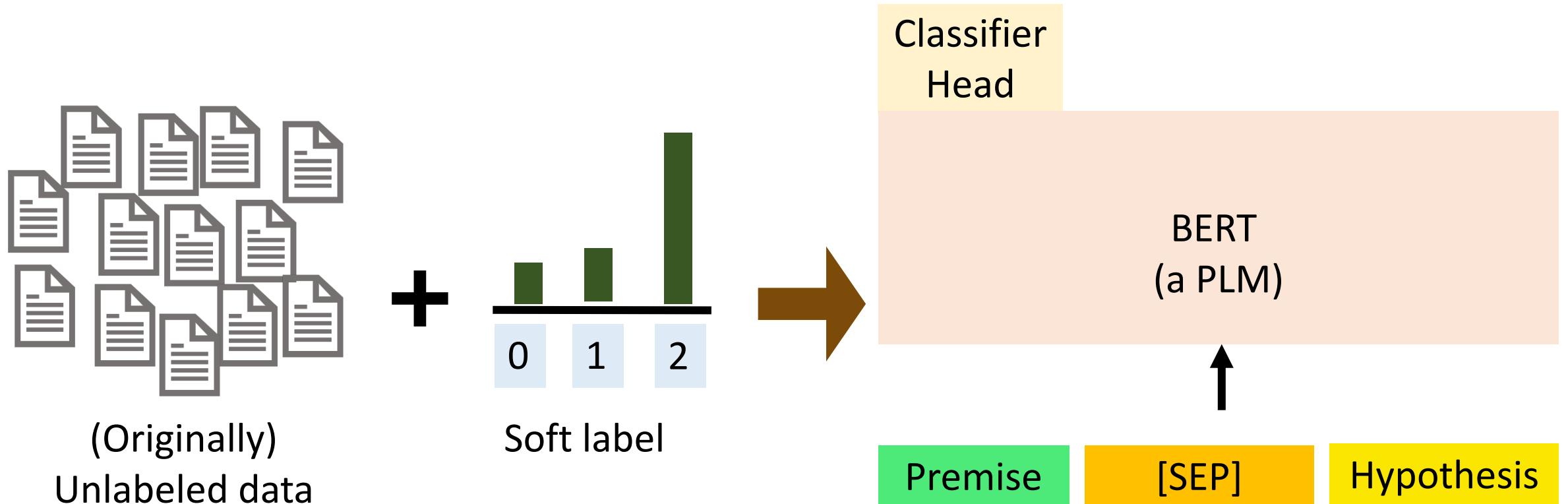
Semi-supervised learning with PLMs: PET

- Pattern-Exploiting Training (PET)
 - Step 2: Predict the unlabeled dataset and combine the predictions from different models



Semi-supervised learning with PLMs: PET

- Pattern-Exploiting Training (PET)
 - Step 3: Use a PLM with classifier head to train on the soft-labeled data set



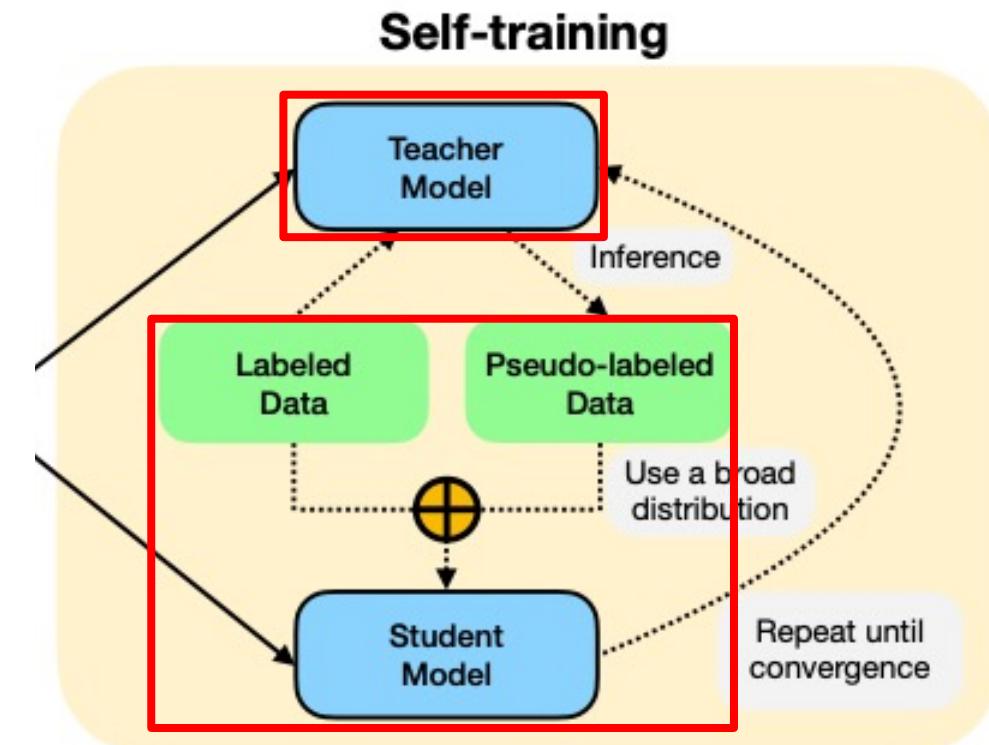
Semi-supervised learning with PLMs: PET

- Pattern-Exploiting Training (PET)
 - Experiment results

	Examples	Method	Yelp	AG's	Yahoo	MNLI (m/mm)
$ \mathcal{T} $: # of labeled samples	\mathcal{T} = 10	supervised	21.1 \pm 1.6	25.0 \pm 0.1	10.1 \pm 0.1	34.2 \pm 2.1 / 34.1 \pm 2.0
		PET	52.9 \pm 0.1	87.5 \pm 0.0	63.8 \pm 0.2	41.8 \pm 0.1 / 41.5 \pm 0.2
	\mathcal{T} = 50	supervised	44.8 \pm 2.7	82.1 \pm 2.5	52.5 \pm 3.1	45.6 \pm 1.8 / 47.6 \pm 2.4
	\mathcal{T} = 100	PET	60.0 \pm 0.1	86.3 \pm 0.0	66.2 \pm 0.1	63.9 \pm 0.0 / 64.2 \pm 0.0
		supervised	53.0 \pm 3.1	86.0 \pm 0.7	62.9 \pm 0.9	47.9 \pm 2.8 / 51.2 \pm 2.6
	\mathcal{T} = 1000	PET	61.9 \pm 0.0	88.3 \pm 0.1	69.2 \pm 0.0	74.7 \pm 0.3 / 75.9 \pm 0.4
		supervised	63.0 \pm 0.5	86.9 \pm 0.4	70.5 \pm 0.3	73.1 \pm 0.2 / 74.8 \pm 0.3
		PET	64.8 \pm 0.1	86.9 \pm 0.2	72.7 \pm 0.0	85.3 \pm 0.2 / 85.5 \pm 0.4

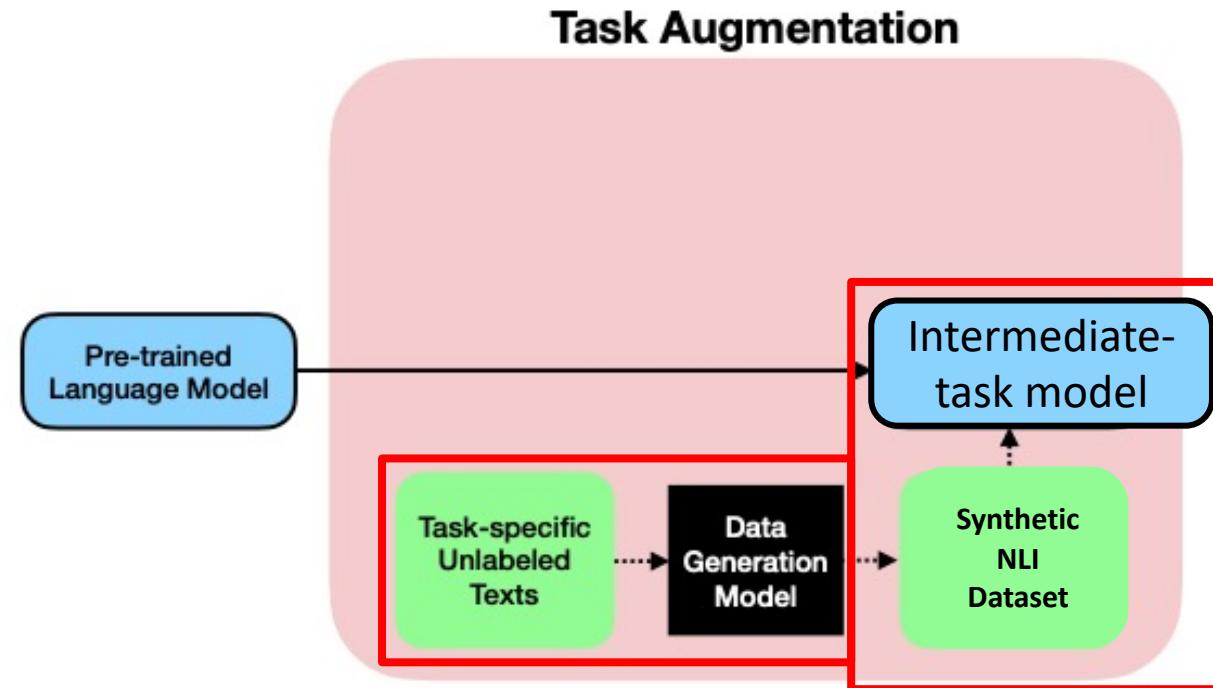
Semi-supervised learning with PLMs: STraTa

- **Self-Training with Task Augmentation (STrATA)**
 - Self-training: use the model's prediction on the unlabeled dataset as pseudo-label
 - How to initialize the models is critical to the performance



Semi-supervised learning with PLMs: STraTa

- **Self-Training with Task Augmentation (STrATA)**
 - Task augmentation: use unlabeled data to generate an NLI dataset, and fine-tuned on the NLI dataset as the intermediate task to obtain the base model



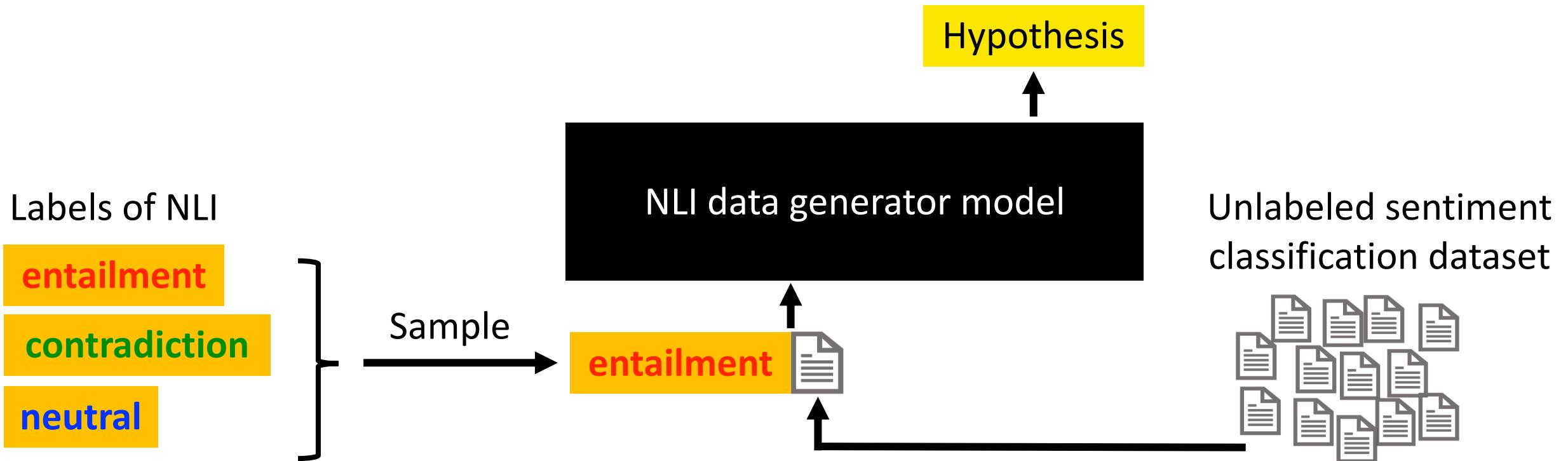
Semi-supervised learning with PLMs: STraTa

- **Self-Training with Task Augmentation (STraTA)**
 - Task augmentation: sentiment classification as the target task
 - Step 1: Train an NLI data generator using another labeled NLI dataset using a generative language model



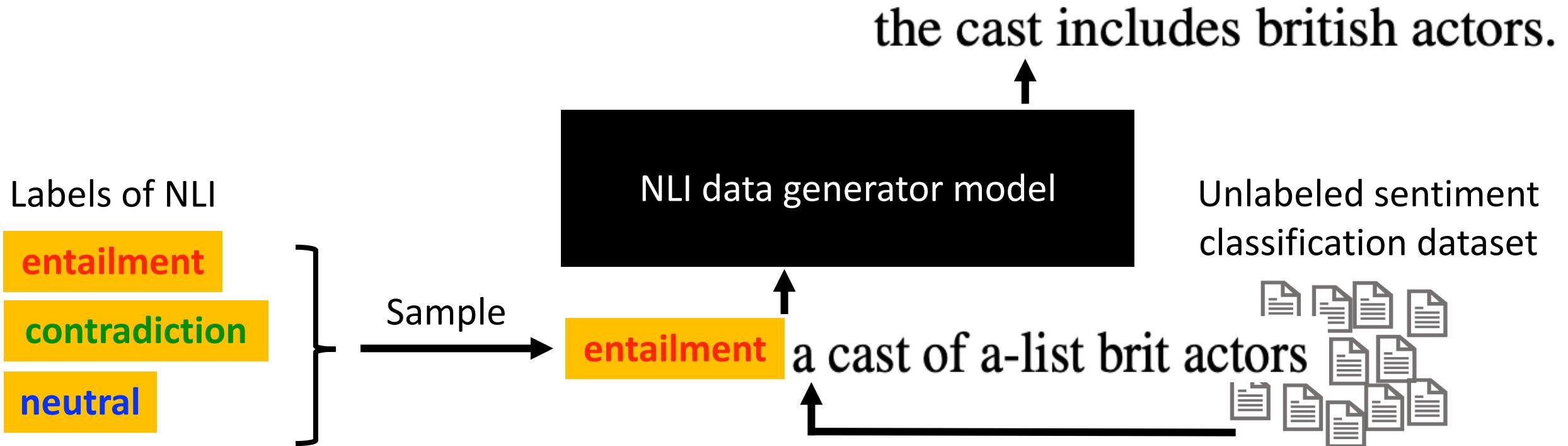
Semi-supervised learning with PLMs: STraTa

- **Self-Training with Task Augmentation (STrATA)**
 - Task augmentation: sentiment classification as the target task
 - Step 2: Use the trained data generator to generate NLI dataset using the in-domain unlabeled data



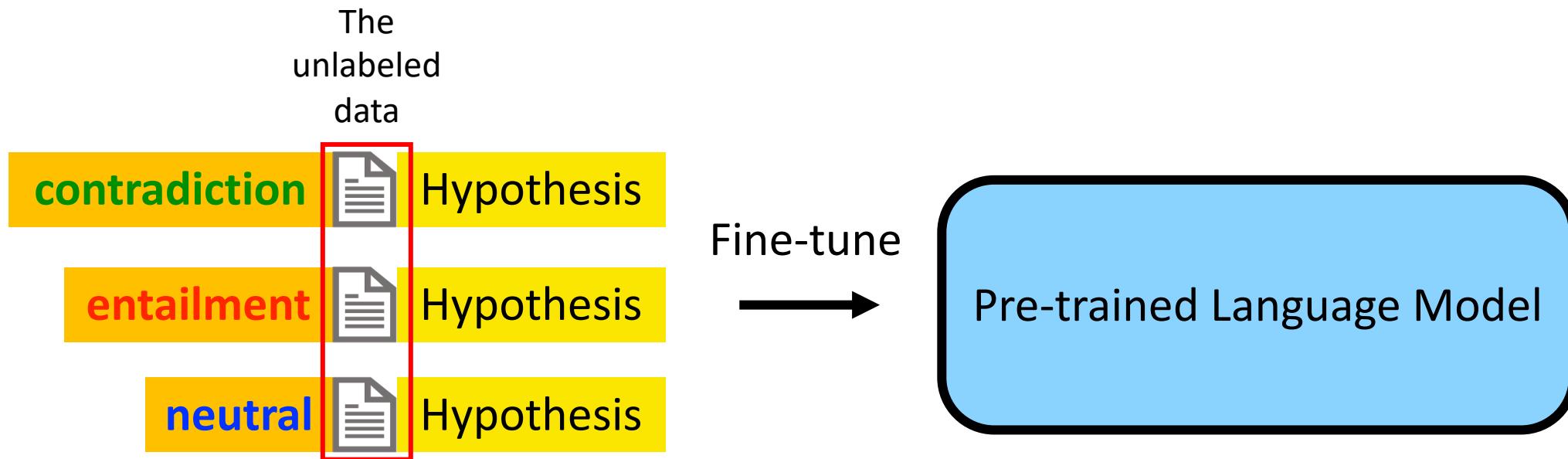
Semi-supervised learning with PLMs: STraTa

- **Self-Training with Task Augmentation (STrATA)**
 - Task augmentation: sentiment classification as the target task
 - Step 2: Use the trained data generator to generate NLI dataset using the in-domain unlabeled data



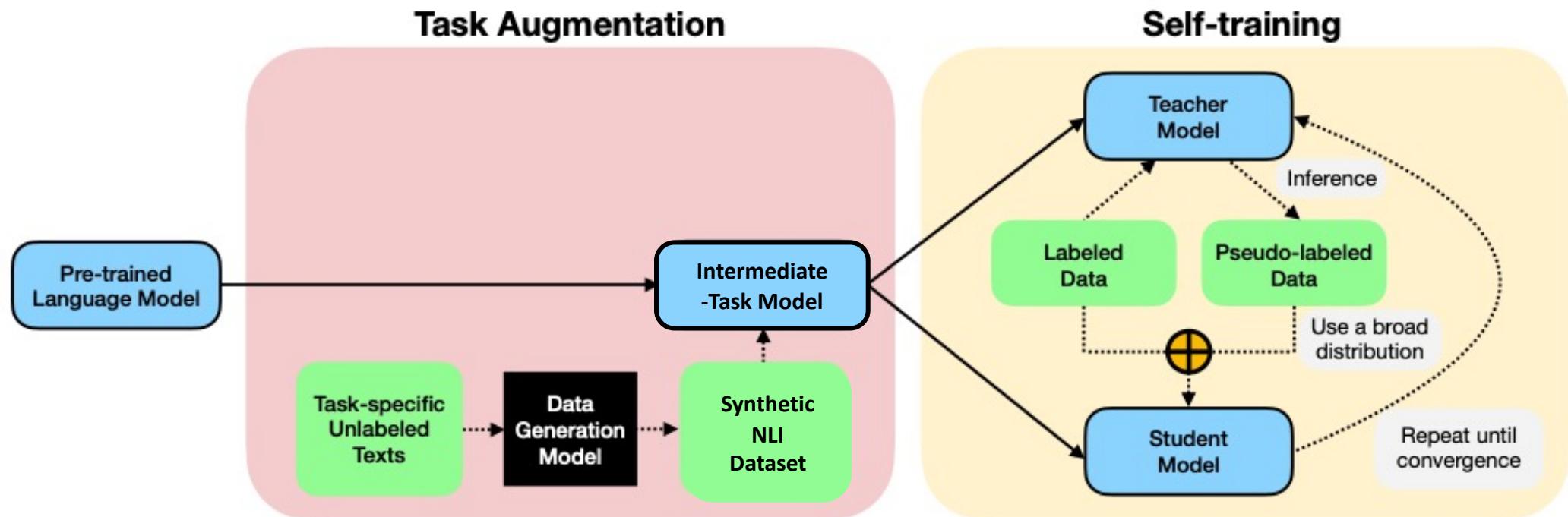
Semi-supervised learning with PLMs: STraTa

- **Self-Training with Task Augmentation (STrATA)**
 - Task augmentation: sentiment classification as the target task
 - Step 3: Use the generated in-domain NLI dataset to fine-tune an NLI model. The fine-tuned model is used to initialize the teacher model and student model in self-training



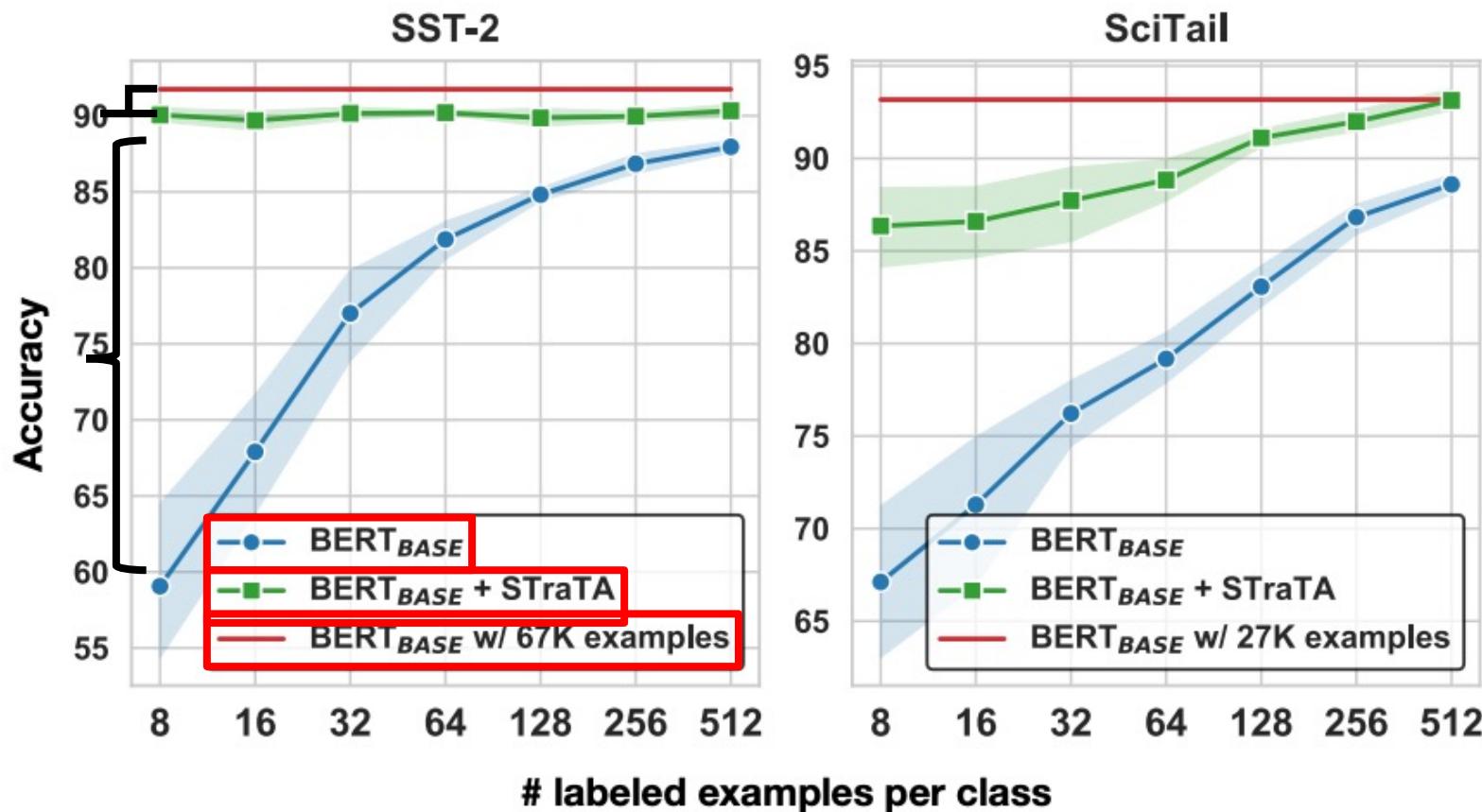
Semi-supervised learning with PLMs: STraTa

- **Self-Training with Task Augmentation (STrATA)**
 - Task augmentation: using sentiment classification as an example
 - Step 3: Use the generated in-domain NLI dataset to fine-tune an NLI model. The fine-tuned model is used to initialize the teacher model and student model in self-training



Semi-supervised learning with PLMs: STraTa

- Self-Training with Task Augmentation (STrTA)





Part 5:

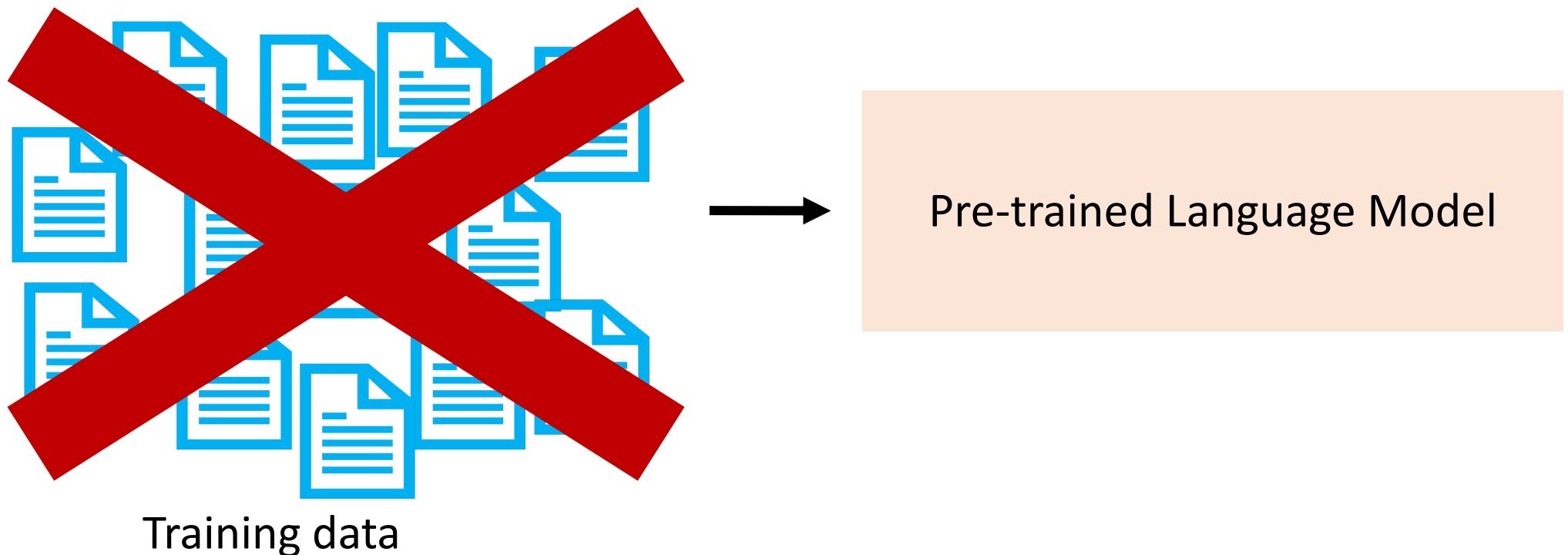
How do PLMs work:

Using PLMs with different amounts of data

5-4: Zero-shot learning

Zero-shot learning

- Zero-shot inference: inference on the downstream task without any training data
- If you don't have training data, then we need a model that can zero-shot inference on downstream tasks

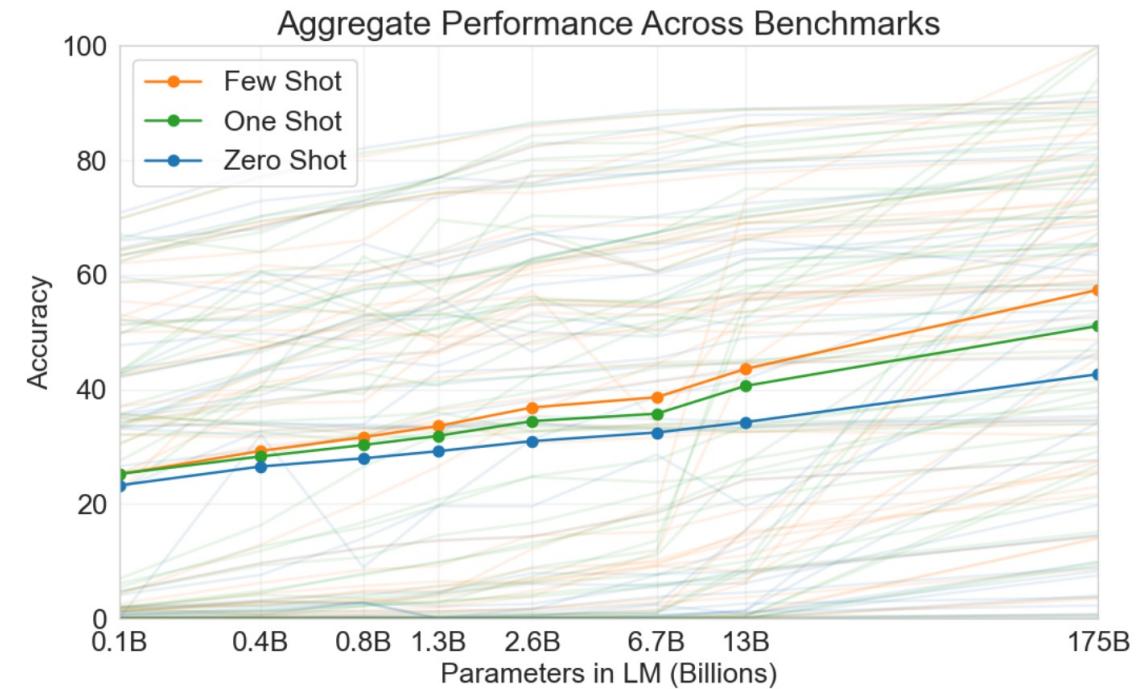
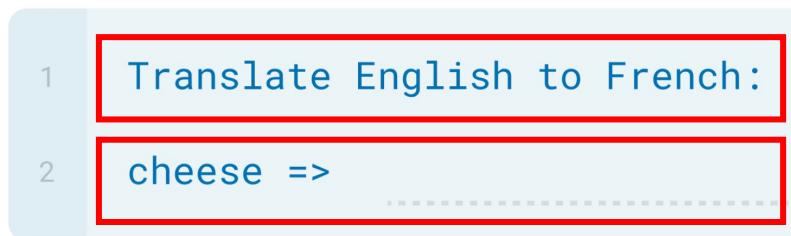


Zero-shot learning

- GPT-3 shows that zero-shot (with task description) is possible

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



Zero-shot learning

- Question: Where does this zero-shot ability spring from?
- Hypothesis: during pre-training, the training datasets implicitly contains a mixture of different tasks
 - QA

Q: I got 4 papers. Should I expect this load in the future?

A: The average monthly load for reviewers should be much closer to 2, but in certain periods (close to large conferences), it's possible that the load is higher.

- Summarization

Finetuned Language Models are Zero-Shot Learners

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, Quoc V Le

29 Sept 2021 (modified: 10 Feb 2022) ICLR 2022 Oral Readers: Everyone Show Bibtex Show Revisions

Keywords: natural language processing, zero-shot learning, language models

Abstract: This paper explores a simple method for improving the zero-shot learning abilities of language models. We show that instruction tuning—finetuning language models on a collection of datasets described via instructions—substantially improves zero-shot performance on unseen tasks. We take a 137B parameter pretrained language model and instruction tune it on over 60 NLP datasets verbalized via natural language instruction templates. We evaluate this instruction-tuned model, which we call FLAN, on unseen task types. FLAN substantially improves the performance of its unmodified counterpart and surpasses zero-shot 175B GPT-3 on 20 of 25 datasets that we evaluate. FLAN even outperforms few-shot GPT-3 by a large margin on ANLI, RTE, BoolQ, AI2-ARC, OpenbookQA, and StoryCloze. Ablation studies reveal that number of finetuning datasets, model scale, and natural language instructions are key to the success of instruction tuning.

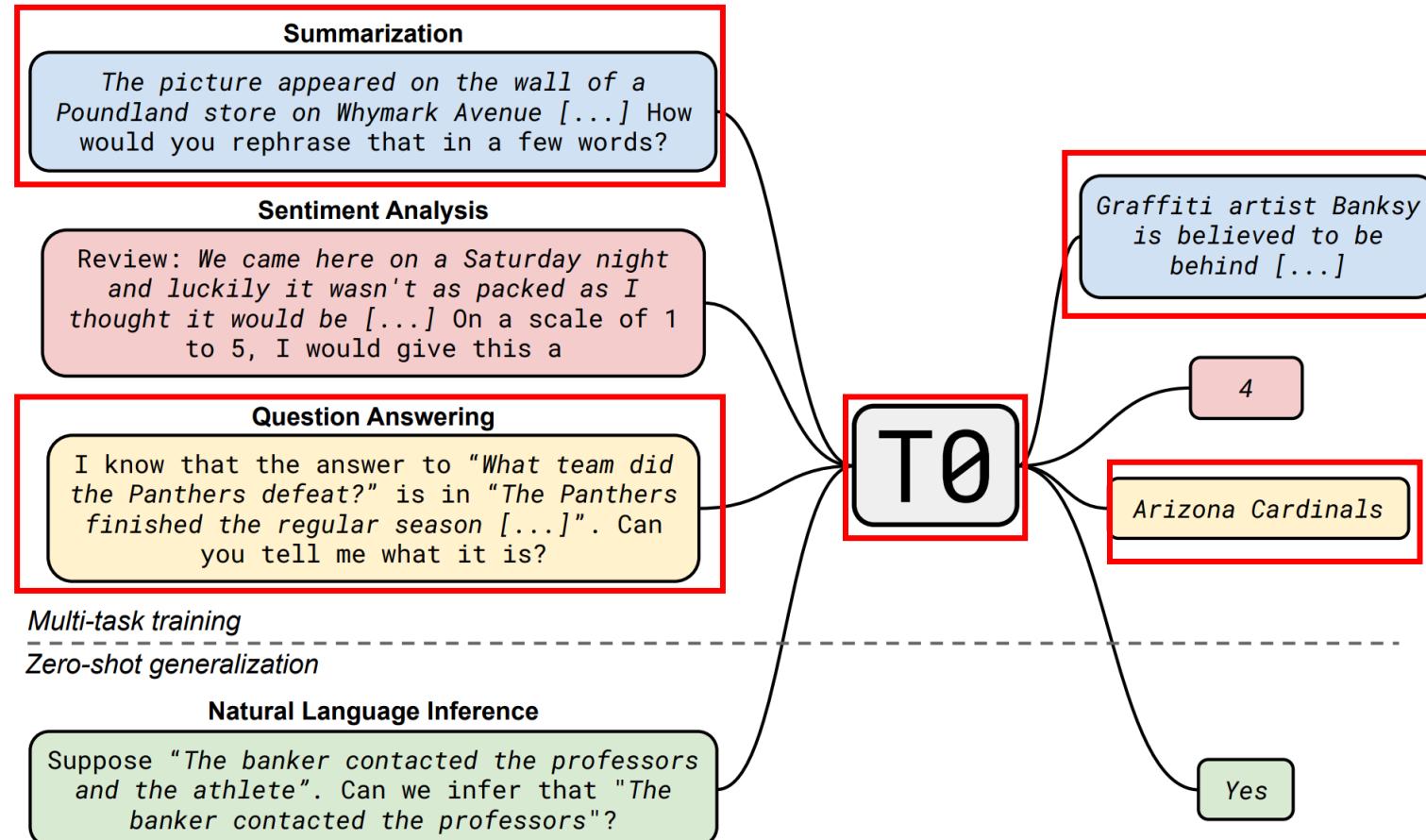
One-sentence Summary: "Instruction tuning", which finetunes language models on a collection of tasks described via instructions, substantially boosts zero-shot performance on unseen tasks.

Zero-shot learning

- Hypothesis: multi-task training enables zero-shot generalization
 - Why not train a model with multi-task learning on a bunch of dataset?

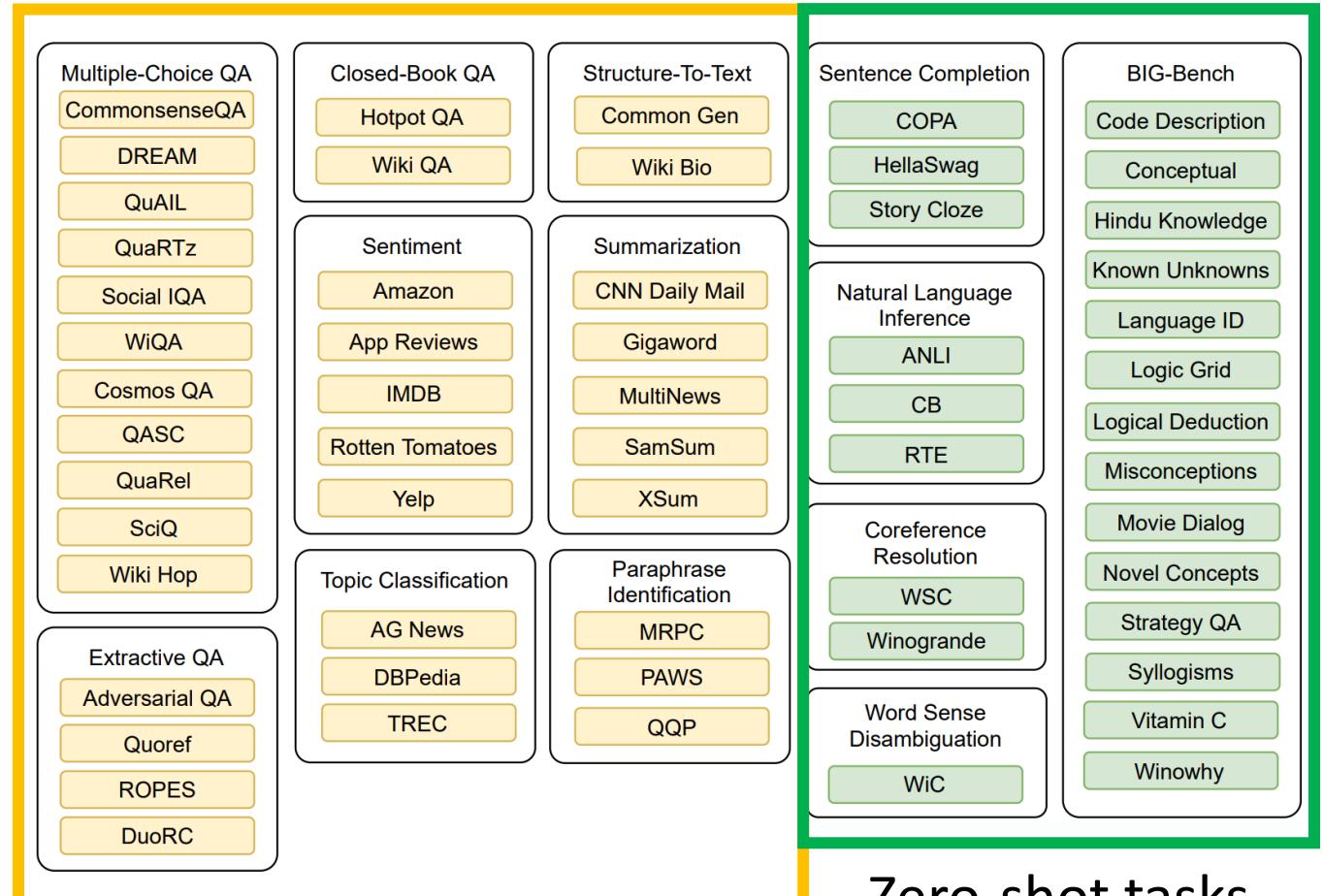
Multi-task
intermediate-task
fine-tuning

Zero-shot
Generalization



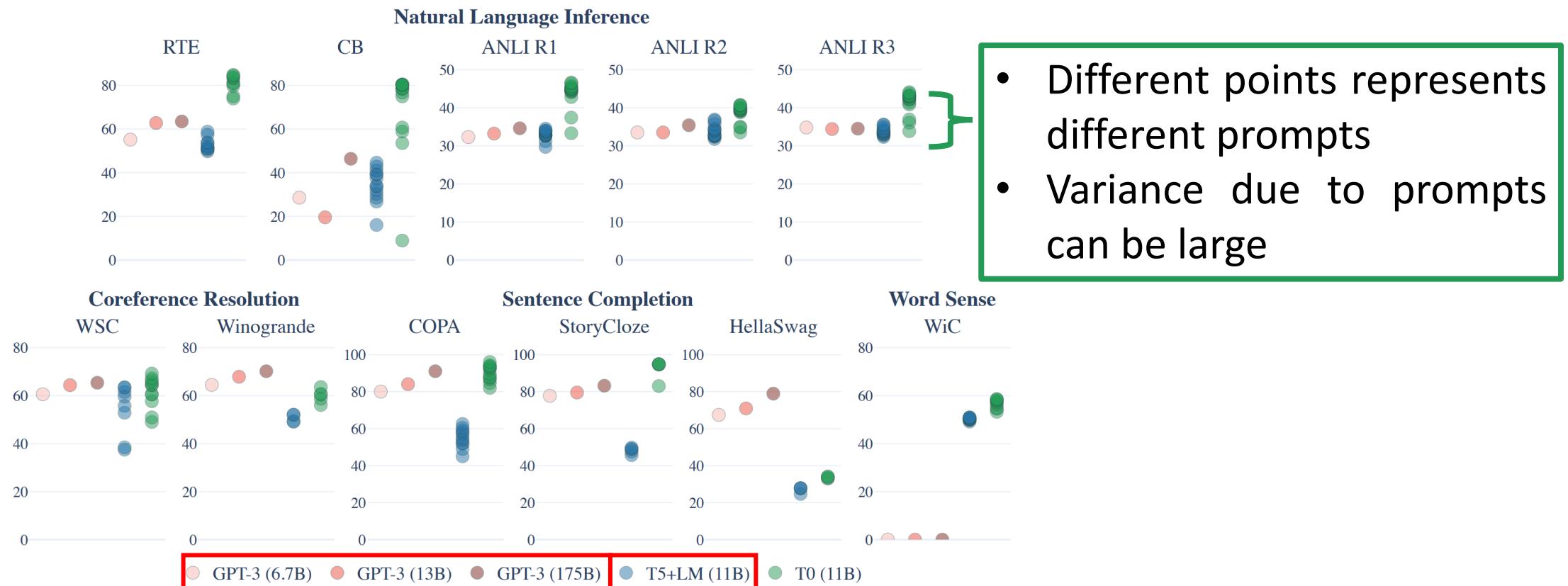
Zero-shot learning

- Intermediate-task fine-tuning with some types of tasks
- Zero-shot inference on other types of tasks



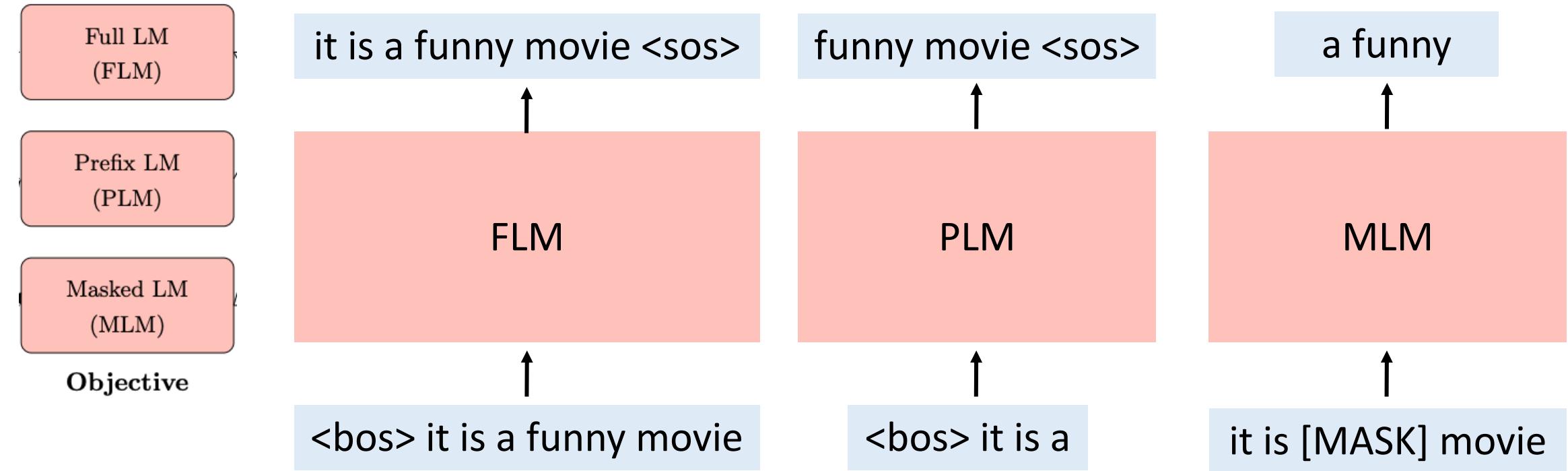
Zero-shot learning

- Sometimes achieves performance better than GPT-3 (175B parameters) with **only 11B** parameters



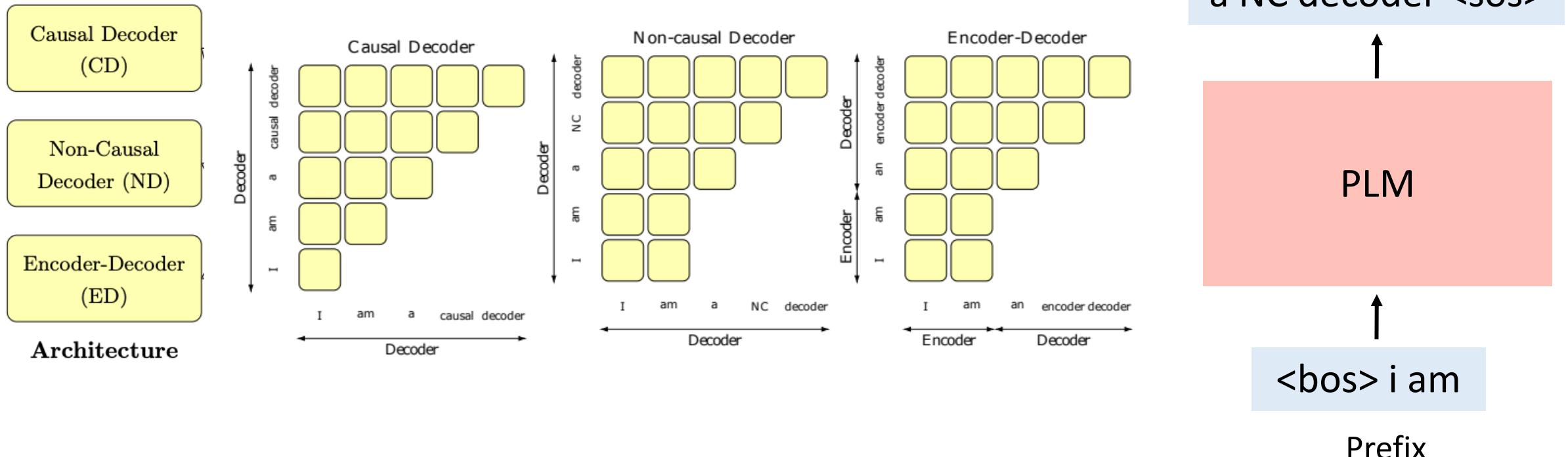
Zero-shot learning

- What language model architecture and pre-training objective work best for zero-shot generalization?



Zero-shot learning

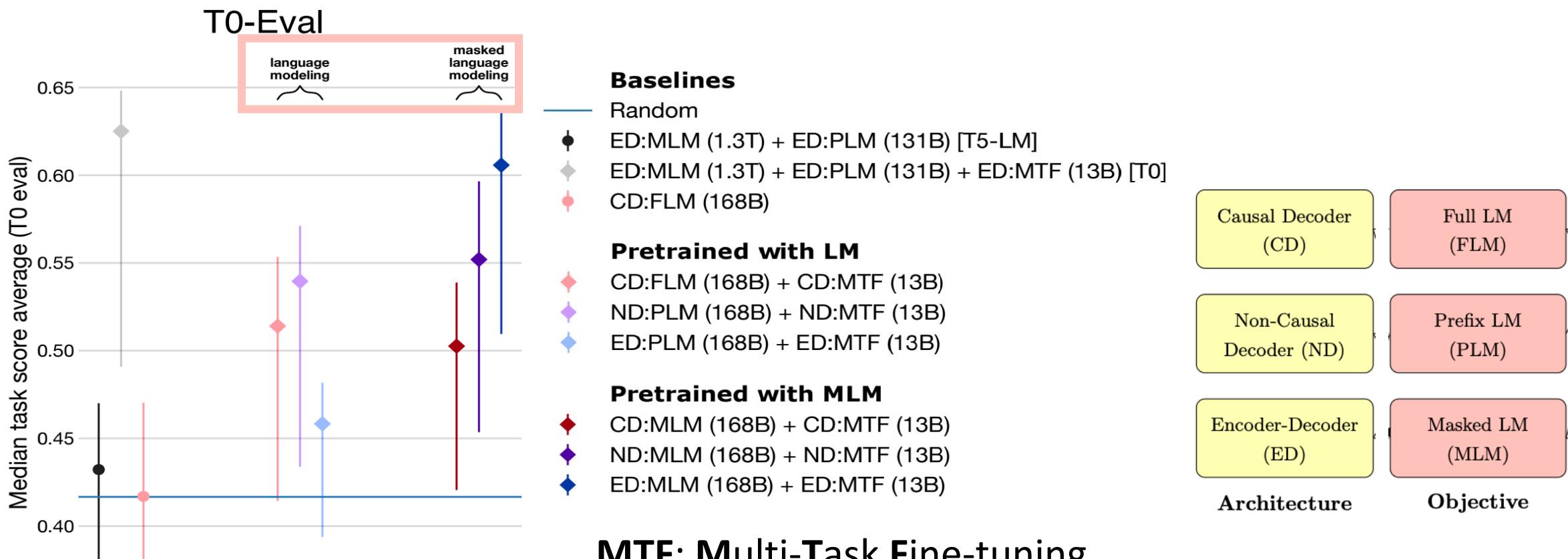
- What language model architecture and pre-training objective work best for zero-shot generalization?



Wang, Thomas, et al. "What Language Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization?." *arXiv preprint arXiv:2204.05832* (2022).

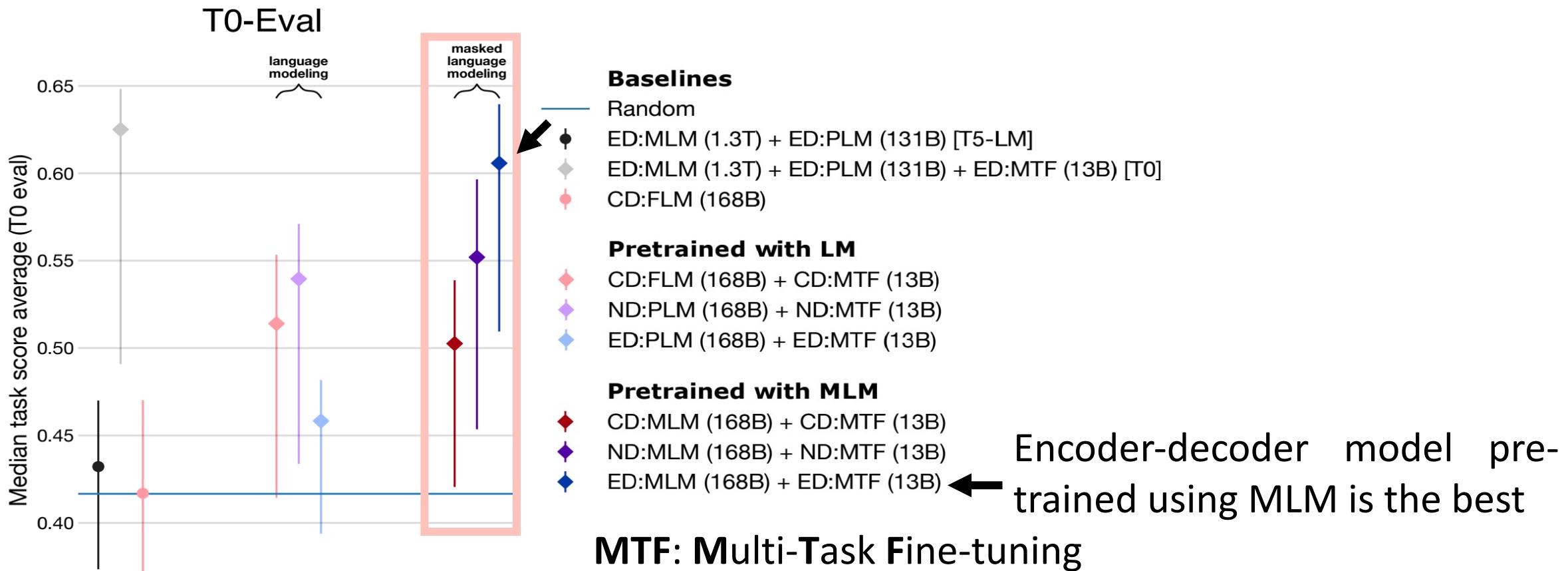
Zero-shot learning

- What language model architecture and pre-training objective work best for zero-shot generalization?



Zero-shot learning

- What language model **architecture** and **pre-training objective** work best for zero-shot generalization?





Part 5:

How do PLMs work:

Using PLMs with different amounts of data

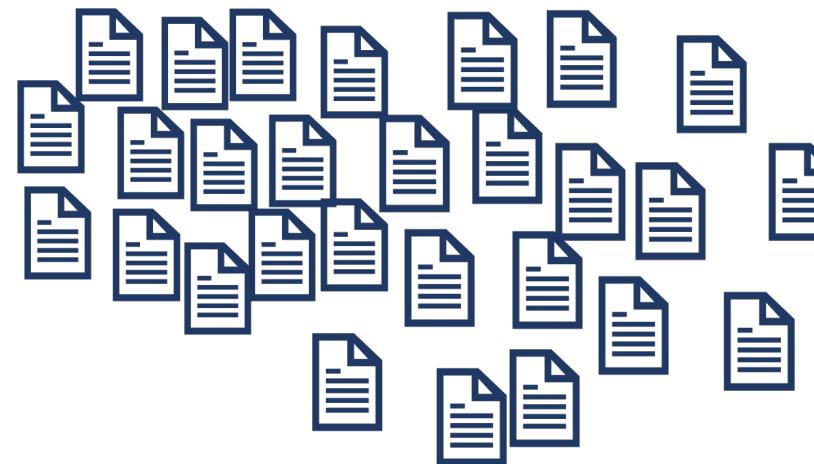
5-5: Short summary

Using PLMs with different amount of data

- PLMs can be used with different amount of labeled and unlabeled data
- Special designs need to be made under different scenarios



Target task dataset
(labeled)



Datasets of other tasks
(labeled)

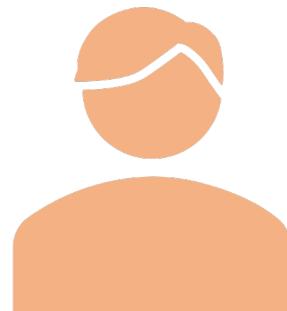


Data related to target task
(Unlabeled)

Using PLMs with different amount of data

- Use natural language prompts and add scenario-specific designs

- [CLS] The spring break is coming soon. Is it true that the spring break was over? >>> **no**
- [CLS] I am going to have dinner. Is it true that I am going to eat something? >>> **yes**
- [CLS] Mary likes pie. Is it true that Mary hates pie. [SEP]
>>> ?



yes



Schedule

- 17:00 – 17:10 **Part 1** Introduction [Hung-yi]
- 17:10 – 17:40 **Part 2** Why do PLMs work [Hung-yi]
- 17:40 – 18:20 **Part 3** How to use PLMs: Contrastive Learning for PLMs [Yung-Sung]
- 18:20 – 18:30 Q&A for Part 1+2+3
- 18:30 – 18:40 Break
- 18:40 – 19:05 **Part 4** How to use PLMs: Parameter-efficient fine-tuning [Cheng-Han]
- 19:05 – 19:50 **Part 5** How to use PLMs: Using PLMs with different amounts of data [Cheng-Han]
- 19:50 – 20:00 Conclusion and Future work + Q&A



2022 AACL-IJCNLP

Conclusion and Future Work + Q&A

Conclusion

- Researchers have studied why PLMs are useful from many aspects
- Contrastive learning is a powerful method to obtain high quality sentence embedding in an unsupervised way
- Parameter-efficient fine-tuning can achieve comparable performance to full-model fine-tuning
- PLMs can be used in with different amount of labeled and unlabeled datasets, and incorporating human knowledge is very critical the performance

Future work

- Why PLMs work is not completely answered yet, including the mathematical theory / learning theory behind the PLMs
- How can we create better negative and positive samples for contrastive learning in an unsupervised way?
- How can we combine parameter-efficient fine-tuning methods with other methods (pruning, compression, quantization) to further reduce the parameters?
- How does those few-shot learning methods perform domain-specific datasets?
- How trust-worthy are the prediction of PLMs, especially in few-shot and zero-shot?

Future work

- Why is the variance between different prompts very large for certain tasks? Does this imply the PLM fail to understand human language?
- How do we continuously adapt PLMs to different domain and datasets from different time?

Still a long way to go!



Recent Advances in PLMs: Why Do They Work and How to Use Them

Part 1 Introduction

Part 2 Why do PLMs work

Part 3 How to Use PLMs: Contrastive learning

Part 4 Parameter-efficient finetuning

Part 5 Using PLMs with different amounts of data

Any questions?