

# Zero- and Few-Shot NLP with Pretrained Language Models

Iz Beltagy, Arman Cohan, Robert L. Logan IV, Sewon Min, Sameer Singh



# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation and benchmarks [Arman]
- 17:05–17:25 Part 6: Meta-training [Arman]
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]

# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation benchmark [Arman]
- 17:05–17:25 Part 6: Meta-training [Arman]
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]

# Part 1: Introduction

---

Slides available at:

<https://github.com/allenai/acl2022-zerofewshot-tutorial>

# What is Few-Shot Learning?

“Learning a task with minimal task description”

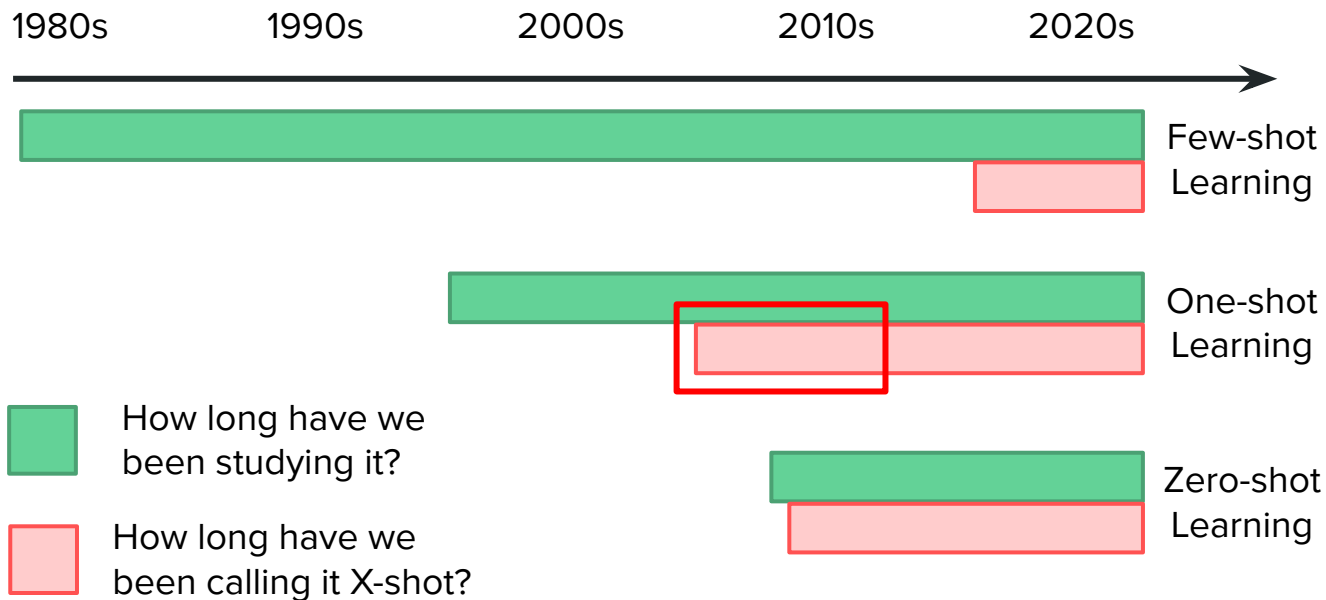
Task description??

- Input and outputs
- Representing task as a prompt
- Instructions on what it is



Expectation of efficiency  
In memory and speed

# History of Zero/One/Few-Shot Learning\*



One-shot means  
1-example per class?



**Nope, it was mostly used in the “learning in one shot” sense!**

\*I am neither a historian, nor *that* old

# Why do we care about Few-Shot Learning?

Practically Useful

Scientifically Useful

# Practically Useful

## Labeling data is costly

- Requires domain expertise
  - Medical, legal, financial
- Inputs are long/complex
  - Grammaticality
- Output is not a class
  - Semantic parsing



# Practically Useful

Labeling data is costly

**You want to do best with what you have**

- You don't want to get more data
- “Cold-start” Recommendations
  - I saw 3 movies, suggest others
- Misinformation/Factuality Detection
  - Something new happened
  - Need to react quickly!

# Practically Useful

Labeling data is costly

You want to do best with what you have

**Finetuning can unstable**

- Training is sensitive to hyperparams
- Not enough validation data
- Difficult to trust the model works

# Practically Useful

Labeling data is costly

You want to do best with what you have

Finetuning can be unstable

**Finetuning large LMs is expensive**

- Large models are much more accurate
- Expensive to train, time and memory
- Some methods will scale to more tasks
  - hundreds/thousands of tasks

# Scientifically Useful

## Potential test for “Intelligent Behavior”

- Generalization from few examples
  - Fundamental piece of intelligence
  - Often used in psychology
- Version of “Thinking Fast”
  - Quickly adjust to environment

# Scientifically Useful

Potential test for “Intelligent Behavior”

**“But... Deep learning is data hungry!”**

- Long-standing criticism of DL
- Understand why it doesn't work here
  - Or does it?
- What are the new limitations of DL?

# Scientifically Useful

Potential test for “Intelligent Behavior”

“But... Deep learning is data hungry!”

## Insights into Language Modeling

- Evaluating Language Models
  - Do we really care about perplexity?
- What does an LLM “know”?
  - Different Notion of Generalization
- What are the biases/limitations of LLMs?

# Scientifically Useful

Potential test for “Intelligent Behavior”

“But... Deep learning is data hungry!”

Insights into Language Modeling

**Because LARGE language models**

- Training/inference/access is tough
- What else can we do? ;)

# Few-shot classification in other domains

Image classification: identify unseen objects

Robotics: quickly adapt to new environments, imitation learning

Recommendations: suggest good recommendations for new users/items

Graphs: only a few edges/nodes are labeled, figure the rest out

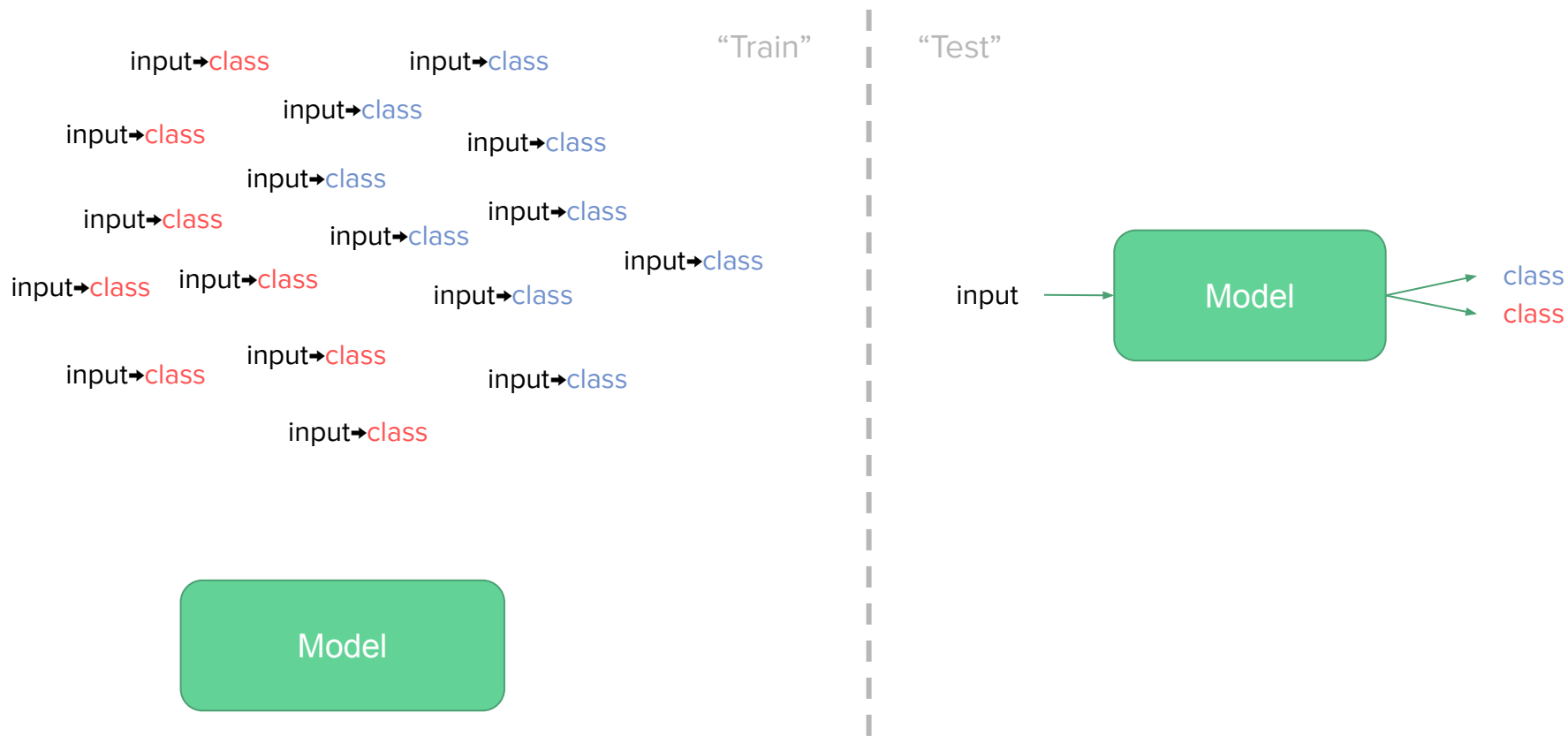
Drug Discovery: quickly identify toxic vs useful drugs

Architecture Search: quickly figure out which hyperparameters to avoid

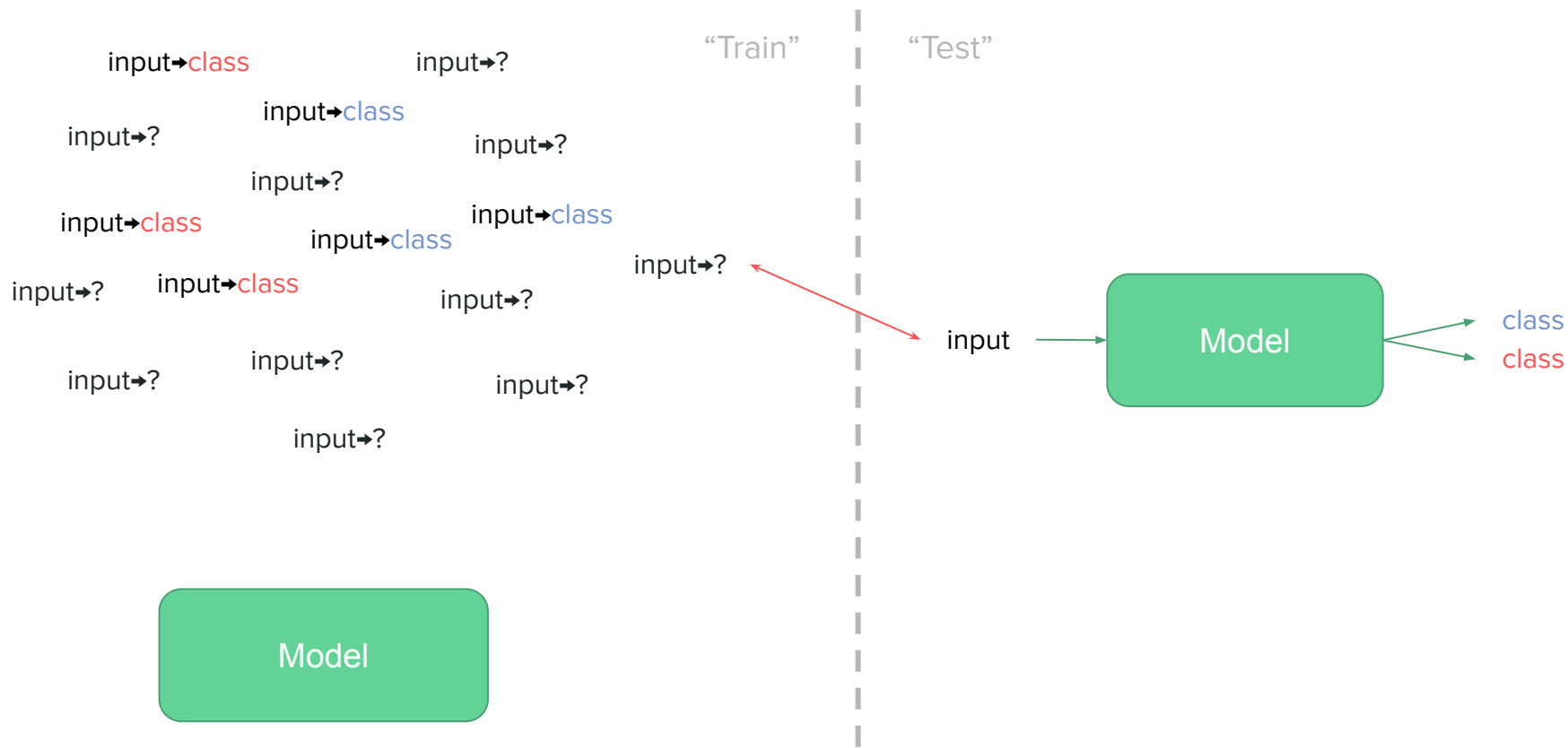
And many others...



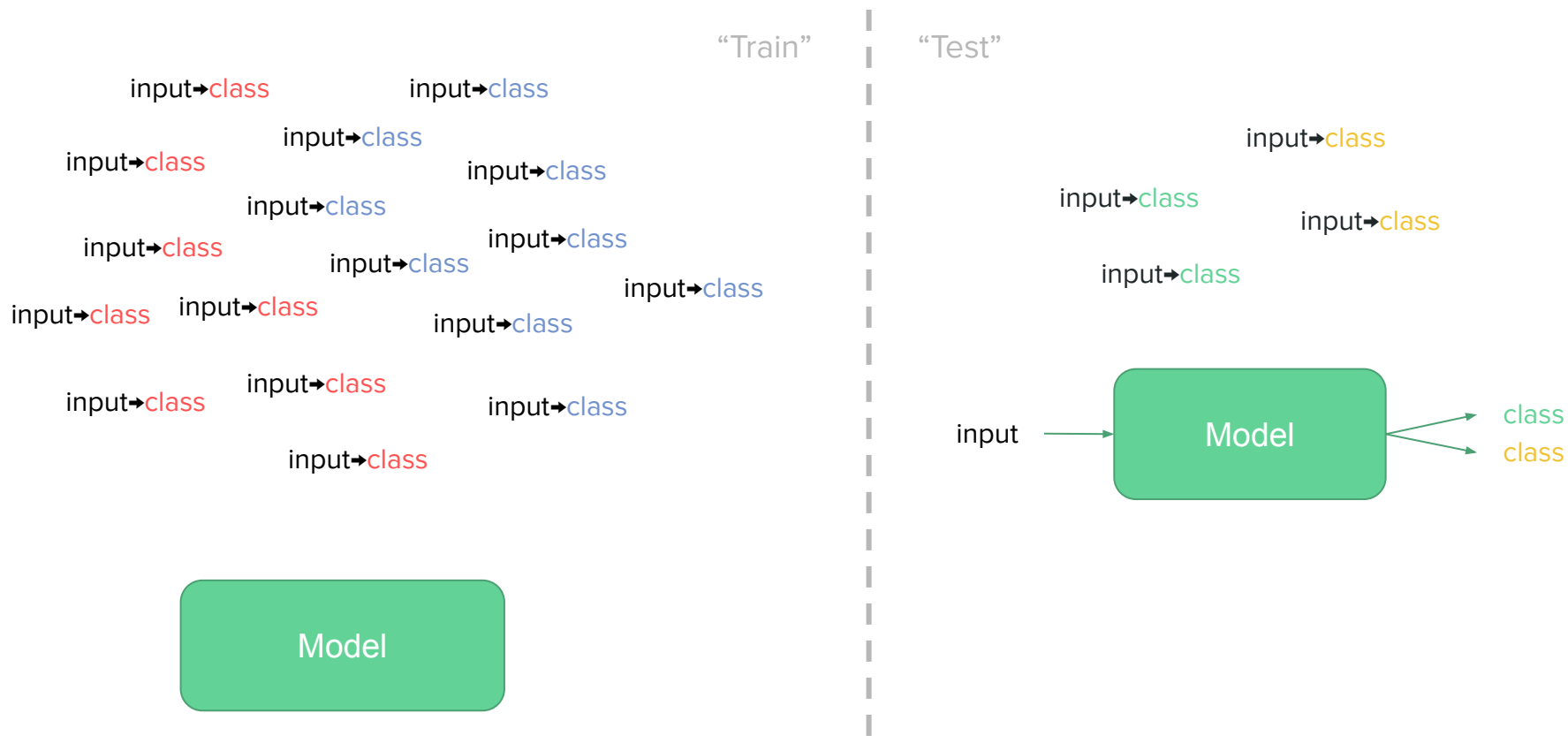
# Related Ideas: Supervised Learning



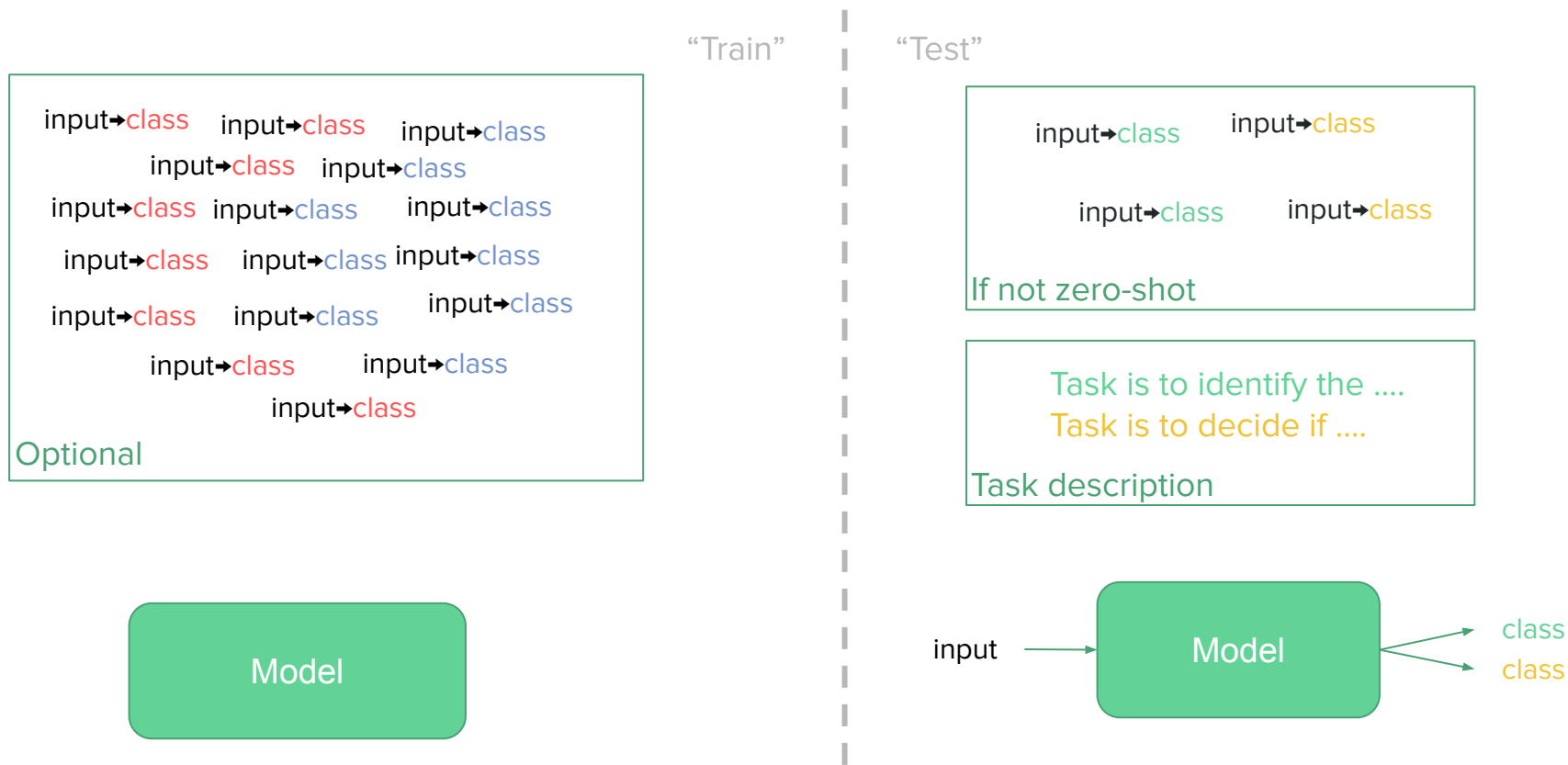
# Related Ideas: Semi-Supervised Learning



# Related Ideas: (traditional) Few-shot learning



# Related Ideas: (modern) Few-shot learning



# What we will <sup>not</sup> cover

- Learning for Low-resource Languages
  - ACL 2022 Tutorial on Learning with Limited Text-Data ([link](#))

Diyi Yang, Ankur P Parikh, Colin Raffel

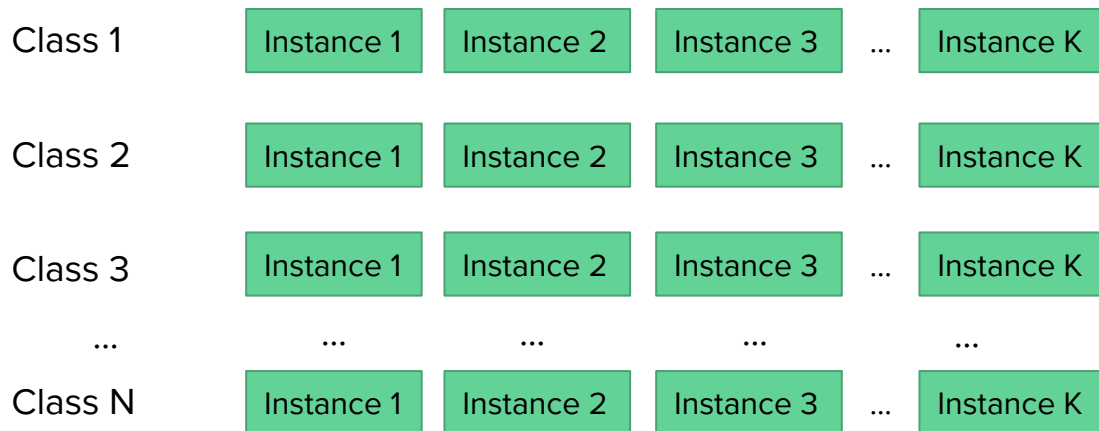


- Our tutorial is primarily focused on English
- Methods for Learning with Limited Data
  - Active Learning, Human-in-the-Loop ML
  - Semi-supervised Learning, Co-training
  - Distant/Weak Supervision, Data Augmentation
  - Transfer learning, Domain adaptation (will touch on this)
  - Learning with Imbalanced classes

# What we will <sup>not</sup> cover

- Model Architectures customized for few-shot learning
  - Mostly used in computer vision for few-shot classification
  - Prototypical Networks
  - Siamese and Matching Networks
- Multiple modalities
  - Lot of relevant work in computer vision and robotics
  - In particular, how tasks can be described using text
- Few-shot Generation Tasks, *specifically*
  - Ideas may work for generation, but not designed for it
- Comprehensive coverage of the related work
  - This is an introduction to the concepts, not a survey
  - E.g. we might mention one meta-learning approach, rather than all of them

# $N$ -way- $K$ -shot Classification



## Often tough in NLP

- Imbalanced classes
  - Can't control distribution
- Open ended classes
  - E.g. topics
- Select from a context
  - E.g. QA
- Text generation
  - E.g. summarization

For the most part, we will use  $K$  for total labeled examples

# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation and benchmarks [Arman]
- 17:05–17:25 Part 6: Meta-training [Arman]
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]



# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation benchmark [Arman]
- 17:05–17:25 Part 6: Meta-training [Arman]
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]

# Part 2: Prompting & In-context learning

(Learning w/o gradient updates)

---


Slides available at:

<https://github.com/allenai/acl2022-zerofewshot-tutorial>

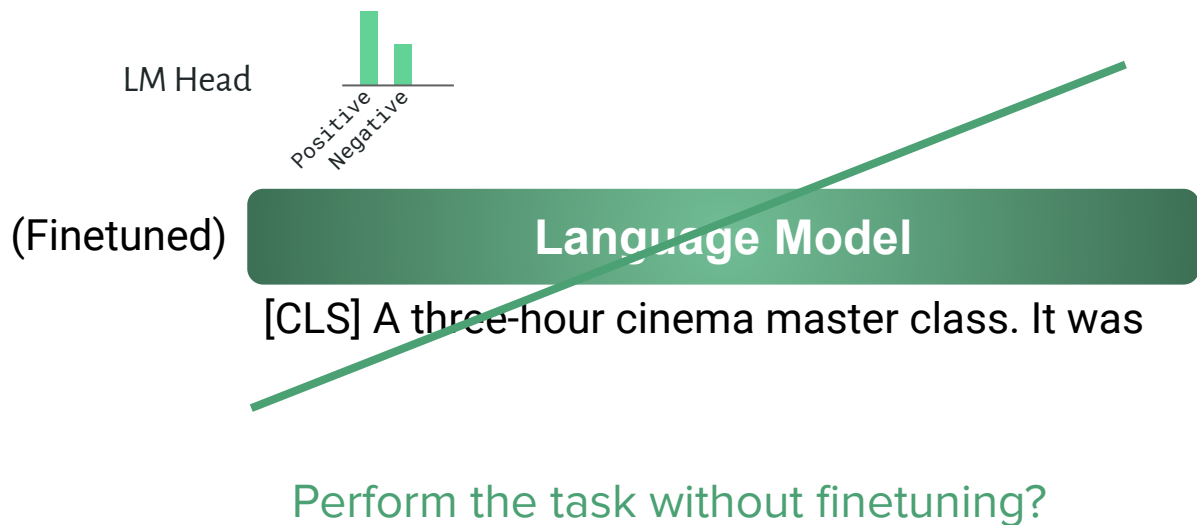
# In this section...

- Language Model Prompting & In-context learning (GPT-3 style)
- Improving in-context learning
  - Better calibration
  - Better scoring of model outputs
  - Better choice of demonstrations
  - Better ordering of demonstrations
- Understanding in-context learning
- Takeaways
- **Caveat**
  - All methods here use no gradient updates (Section 3 for gradient based methods)
  - All methods here use the language model as it is
  - Section 6 for meta-training or instruction learning

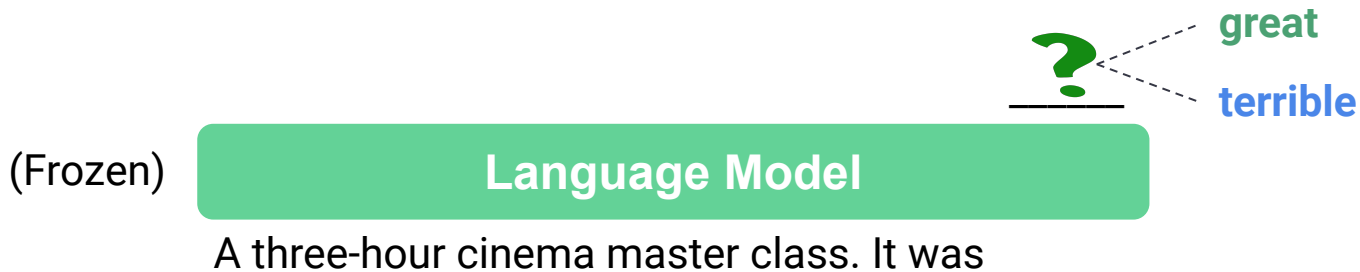
# LM Prompting

A three-hour cinema master class.  **positive**  
**negative**

# LM Prompting



# LM Prompting



P1 = P(It was **great!** | A three-hour cinema master class.)

P2 = P(It was **terrible!** | A three-hour cinema master class.)

P1 > P2    **“positive”**

P1 < P2    **“negative”**

# In-context Learning (GPT3; Brown et al., 2020)

## Movie review dataset

**Input:** An effortlessly accomplished and richly resonant work.

**Label:** positive

**Input:** A mostly tired retread of several other mob tales.

**Label:** negative

An effortlessly accomplished and richly resonant work. It was great!

A mostly tired retread of several other mob tales. It was terrible!

A three-hour cinema master class. It was \_\_\_\_\_

## Language Model

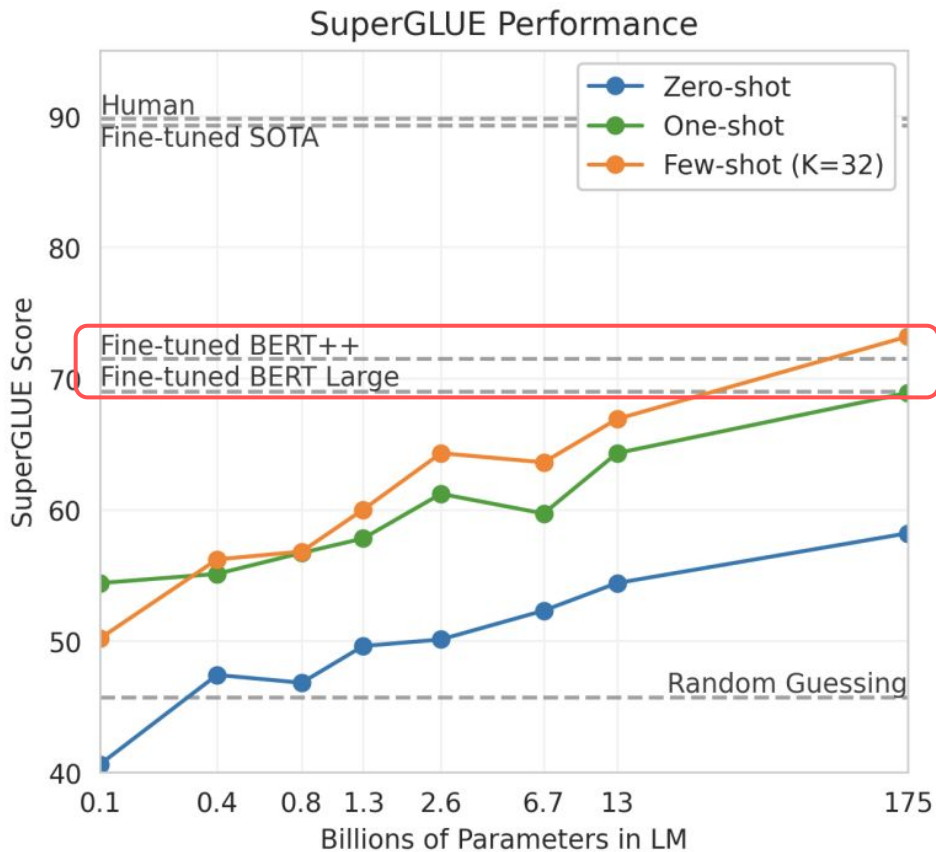
$P1 = P(\text{It was } \text{great!} \mid \text{1st train input+output} \setminus \text{2nd train input+output} \setminus \text{A three-hour cinema master class.})$

$P2 = P(\text{It was } \text{terrible!} \mid \text{1st train input+output} \setminus \text{2nd train input+output} \setminus \text{A three-hour cinema master class.})$

$P1 > P2$     “positive”

$P1 < P2$     “negative”

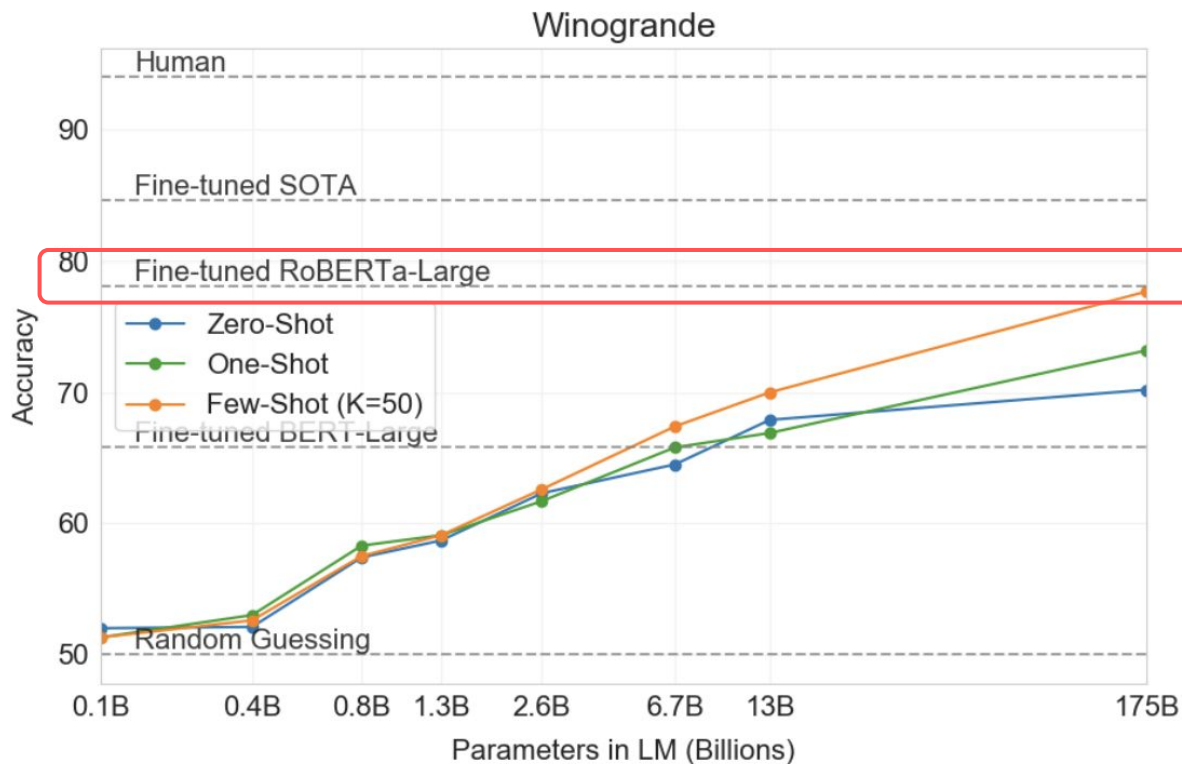
# In-context learning results



Models are Few-Shot Learners”

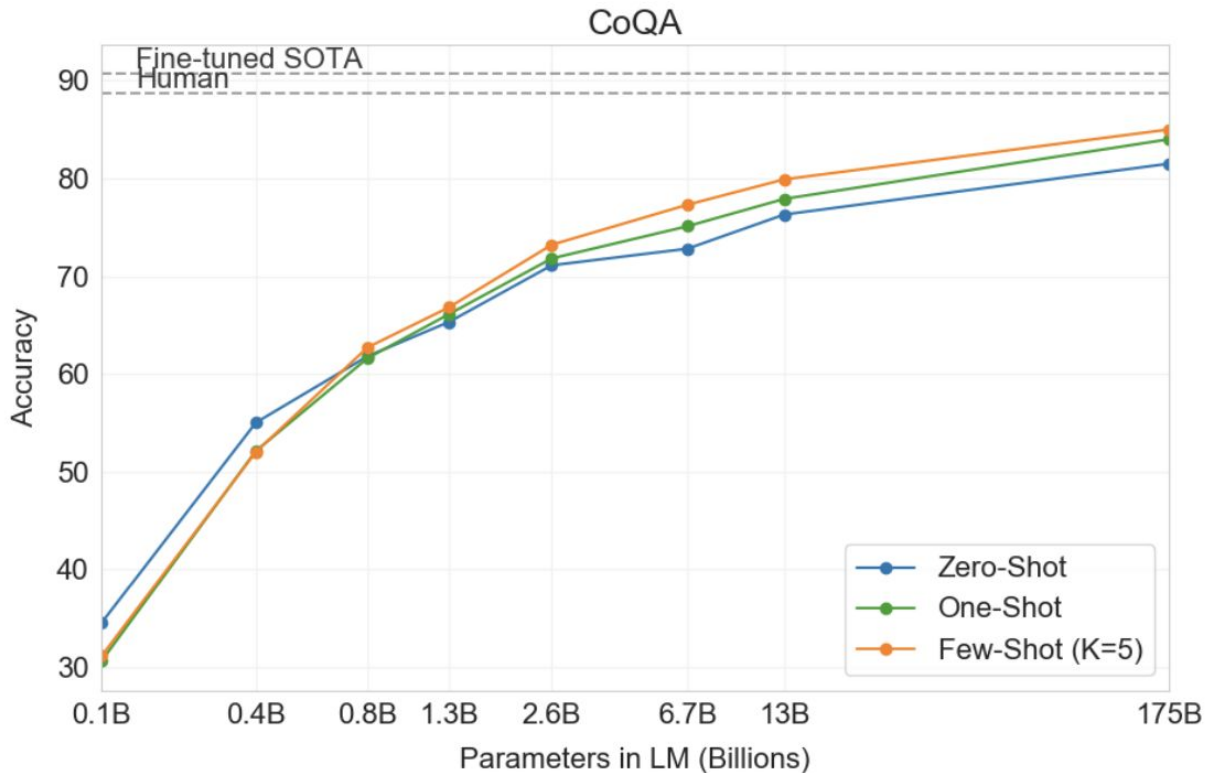


# In-context learning results



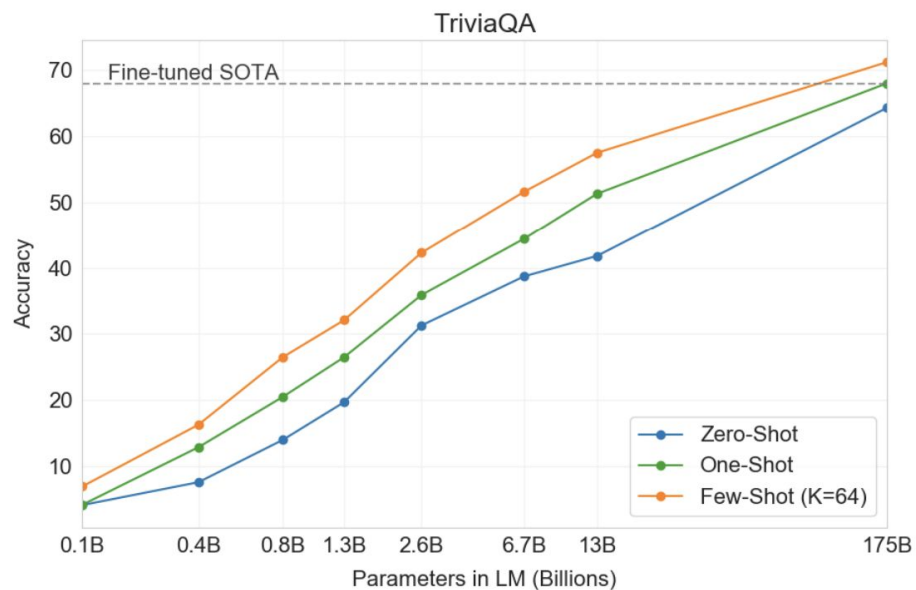
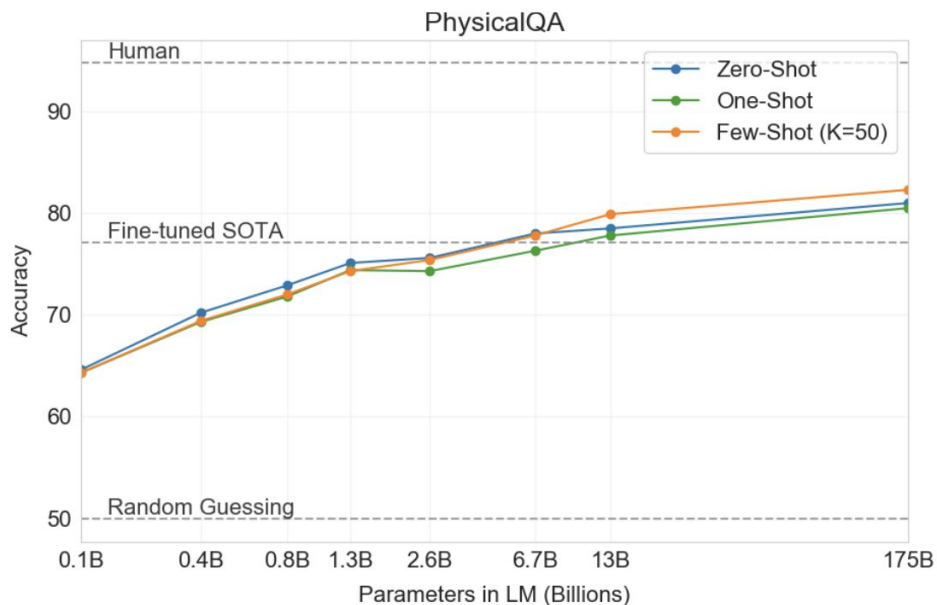
[Brown et al. 2020](#). "Language Models are Few-Shot Learners"

# In-context learning results



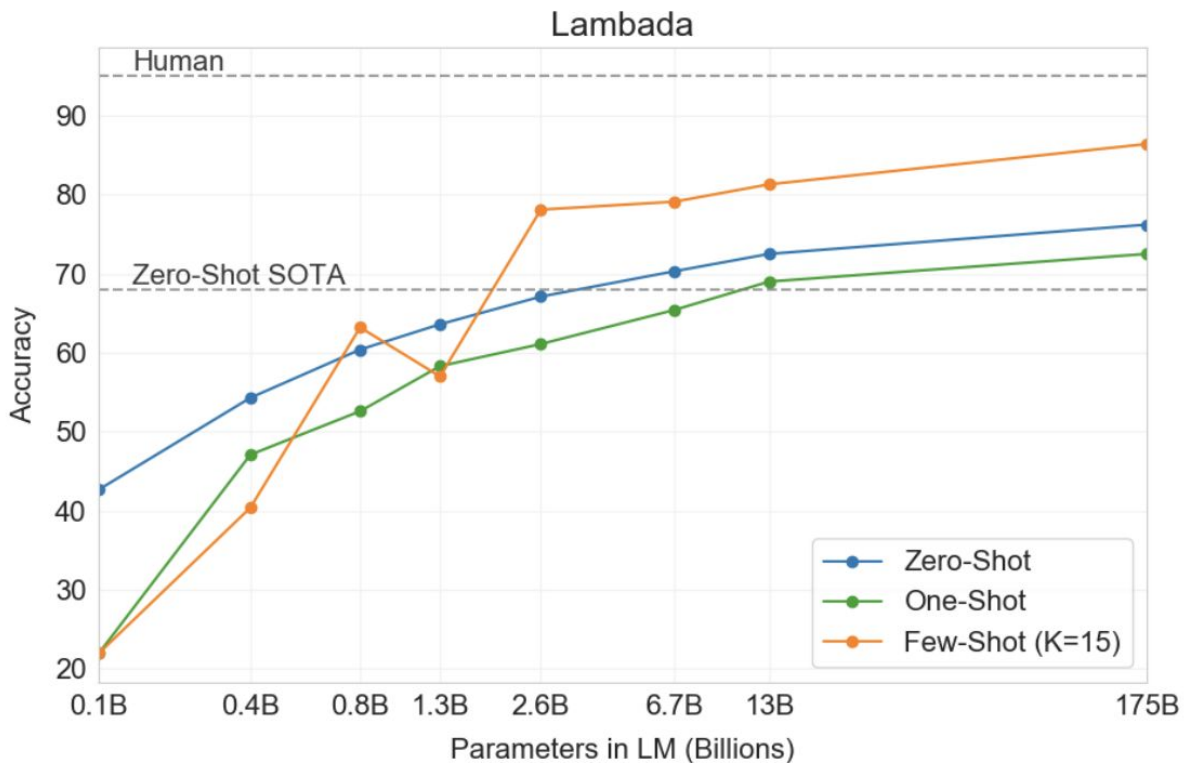
[Brown et al. 2020](#). “Language Models are Few-Shot Learners”

# In-context learning results



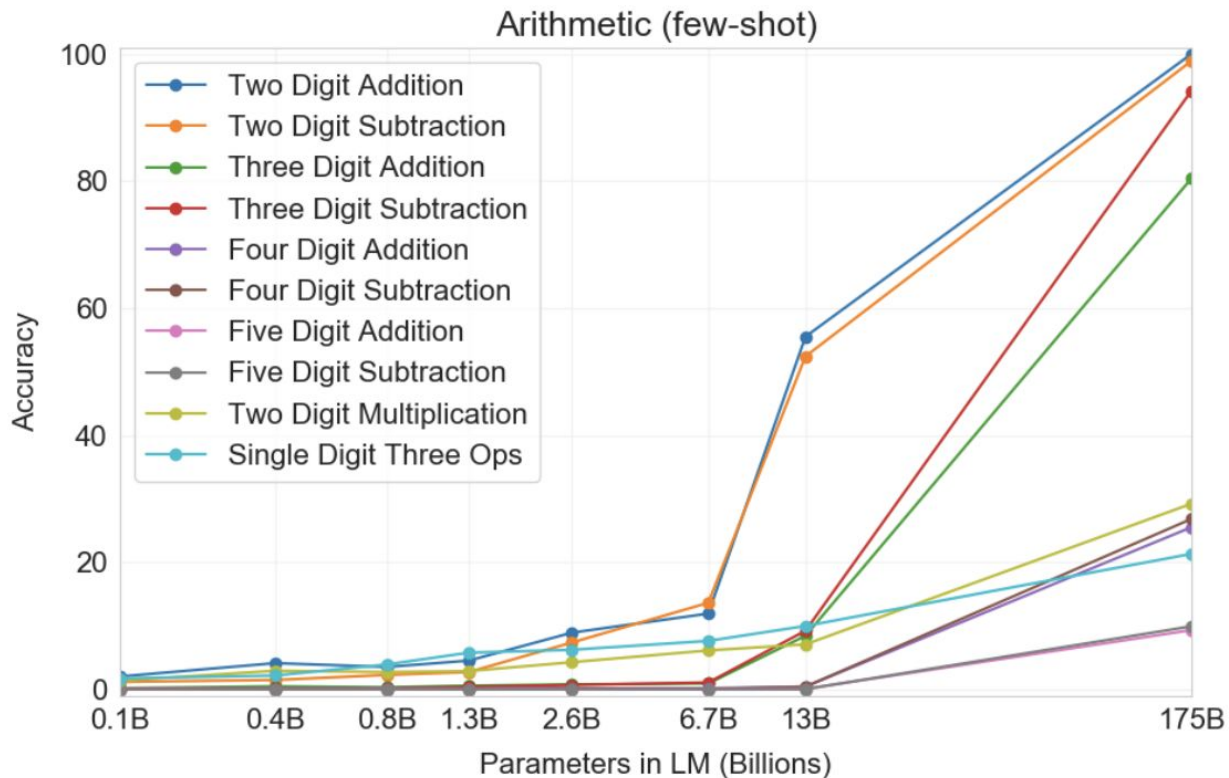
[Brown et al. 2020](#). "Language Models are Few-Shot Learners"

# In-context learning results



Brown et al. 2020. "Language Models are Few-Shot Learners"

# In-context learning results



[Brown et al. 2020](#). “Language Models are Few-Shot Learners”

# Terminologies

# Terminologies

## Input to the LM

An effortlessly accomplished and richly resonant work.	It was great!
A mostly tired retread of several other mob tales.	It was terrible!
A three-hour cinema master class.	It was _____!

**Prompt:** A conditioning text coming before the test input

**Demonstrations:** A special instance of prompt which is a concatenation of the k-shot training data (in in-context learning, prompt==demonstrations)

(In Section 3, prompt can be a series of vectors)

(In the later section, prompt can be a description about the task)

# Terminologies

## Input to the LM

An effortlessly accomplished and richly resonant work.

A mostly tired retread of several other mob tales.

A three-hour cinema master class.

It was great!

It was terrible!

It was \_\_\_\_\_!

**Prompt:** A conditioning text coming before the test input

**Demonstrations:** A special instance of prompt which is a concatenation of the k-shot training data (in in-context learning, prompt==demonstrations)

**Pattern:** A function that maps an input to the text (a.k.a. template)

**Verbalizer:** A function that maps a label to the text (a.k.a. label words)



# Examples of patterns/verbalizers

An effortlessly accomplished and richly resonant work.

It was great!

A mostly tired retread of several other mob tales.

It was terrible!

A three-hour cinema master class.

It was great!

**Pattern:**  $f(\langle x \rangle) = \langle x \rangle$

**Verbalizer:**  $v(\text{"positive"}) = \text{"It was great!"}$ ,  $f(\text{"negative"}) = \text{"It was terrible!"}$

Review: An effortlessly accomplished and richly resonant work.

Sentiment: positive

Review: A mostly tired retread of several other mob tales.

Sentiment: negative

Review: A three-hour cinema master class.

Sentiment: positive

**Pattern:**  $f(\langle x \rangle) = \text{"Review: } \langle x \rangle \text{"}$

**Verbalizer:**  $v(\langle x \rangle) = \text{"Sentiment: } \langle x \rangle \text{"}$

# Notes on patterns/verbalizers

- There are many different possible patterns/verbalizers even for the same task.
- In practice, it is better to use patterns/verbalizers that makes the sequence closer to language modeling, i.e. closer to the text that the model might have seen during pretraining.
- It turns out there is huge variance in performance based on the choice of patterns/verbalizers (more in the next slide).
- You should not choose patterns/verbalizers based on the test data.

# Review

**Test data:**  $(x, y)$    **Train data:**  $(x_1, y_1, \dots, x_k, y_k)$    **Pattern:**  $f$    **Verbalizer:**  $v$

**Zero-shot prompting:**  $\operatorname{argmax}_{y \in \mathcal{Y}} P_{\text{LM}}(v(y) | f(x))$

**In-context learning:**  $\operatorname{argmax}_{y \in \mathcal{Y}} P_{\text{LM}}(v(y) | f(x_1), v(y_1), \dots, f(x_k), v(y_k), f(x))$

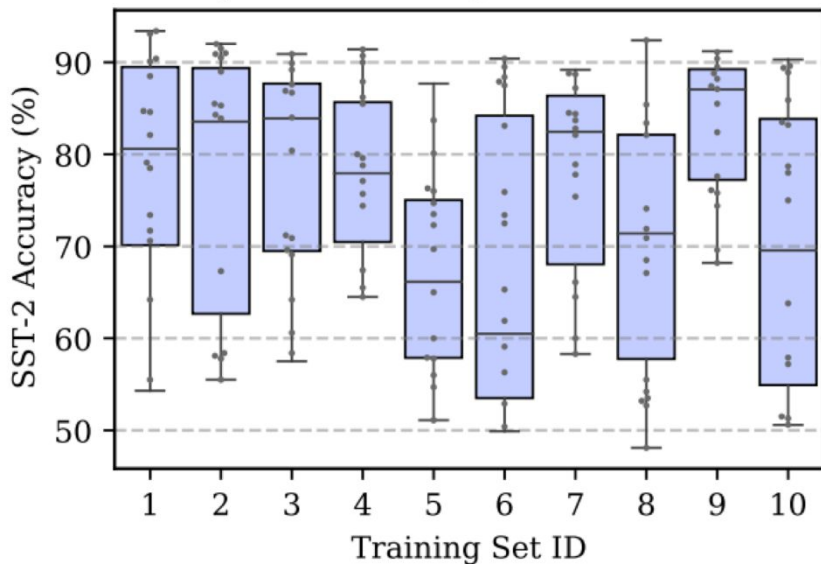
----- **For simplicity, from now on...** -----

**Zero-shot prompting:**  $\operatorname{argmax}_{y \in \mathcal{Y}} P_{\text{LM}}(y | x)$

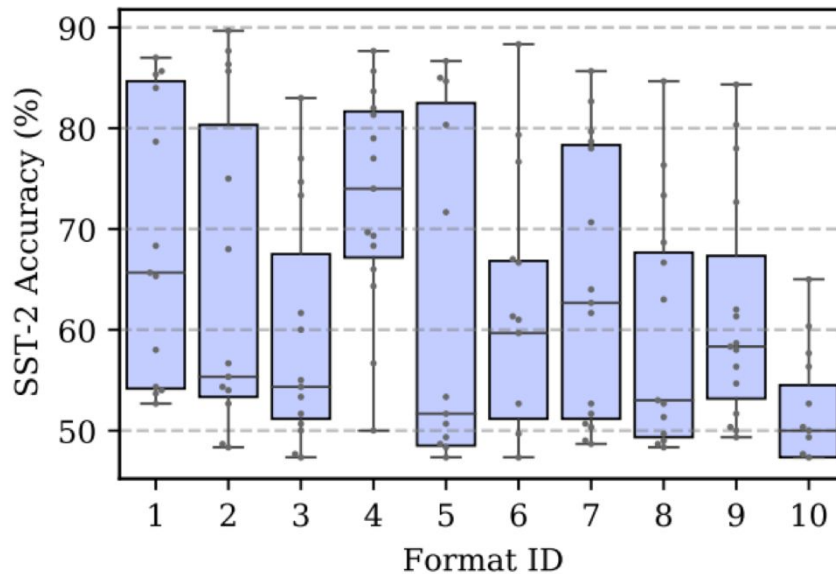
**In-context learning:**  $\operatorname{argmax}_{y \in \mathcal{Y}} P_{\text{LM}}(y | x_1, y_1, \dots, x_k, y_k, x)$

# Variance

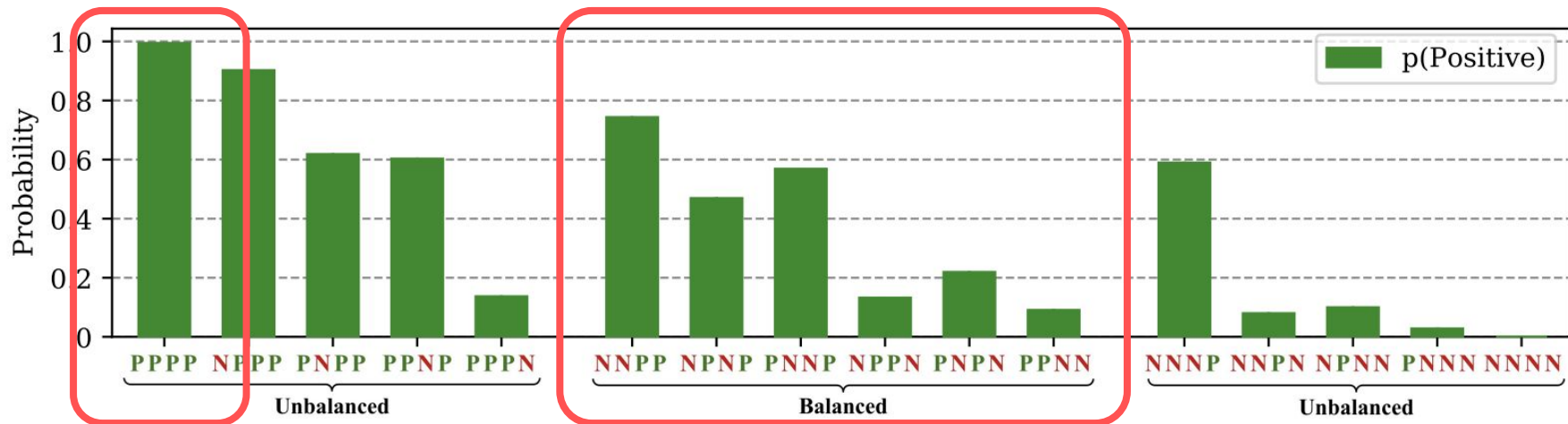
Across different training sets and permutations



Across different training sets and patterns/verbalizers



# Variance



**Figure 4. Majority label and recency biases** cause GPT-3 to become biased towards certain answers and help to explain the high variance across different examples and orderings. Above, we use 4-shot SST-2 with prompts that have different class balances and permutations, e.g., [P P N N] indicates two positive training examples and then two negative. We plot how often GPT-3 2.7B predicts Positive on the balanced validation set. When the prompt is unbalanced, the predictions are unbalanced (*majority label bias*). In addition, balanced prompts that have one class repeated near the end, e.g., end with two Negative examples, will have a bias towards that class (*recency bias*).

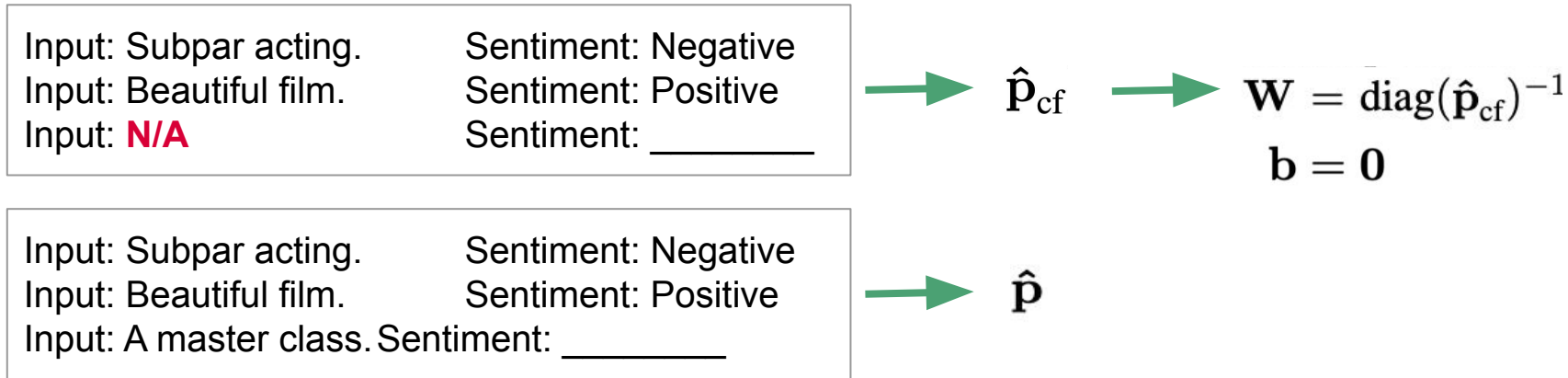
# Calibrate Before Use

- Platt Scaling  $\hat{\mathbf{q}} = \text{softmax}(\mathbf{W}\hat{\mathbf{p}} + \mathbf{b})$

# Calibrate Before Use

- Platt Scaling  $\hat{\mathbf{q}} = \text{softmax}(\mathbf{W}\hat{\mathbf{p}} + \mathbf{b})$
- For zero- or few-shot, we don't have data to learn  $\mathbf{W}$  and  $\mathbf{b}$
- Data-free procedure to infer a good setting:
  - The model's bias toward certain answers can be estimated by feeding in a *content-free* input
  - We can set  $\mathbf{W}$  and  $\mathbf{b}$  so that outputs for the content-free input are *uniform*

# Calibrate Before Use



$$\hat{\mathbf{q}} = \text{softmax}(\mathbf{W}\hat{\mathbf{p}} + \mathbf{b})$$



# Surface Form Competition

A three-hour cinema master class. It was \_\_\_\_\_

Language Model



great

# Surface Form Competition

A three-hour cinema master class. It was \_\_\_\_\_

Language Model



great  
awesome  
excellent  
fantastic  
perfect  
terrific  
wonderful  
exceptional



# Domain Conditional PMI

How much “**more**” likely  $y$  is

$$\text{PMI}(\mathbf{x}, \mathbf{y}) = \log \frac{P(\mathbf{y}|\mathbf{x})}{P(\mathbf{y})} \longleftarrow \text{Unconditional probability is poorly calibrated!}$$

$$\begin{aligned} \text{PMI}_{\text{DC}}(\mathbf{x}, \mathbf{y}, \text{domain}) &= \frac{P(\mathbf{y}|\mathbf{x}, \text{domain})}{P(\mathbf{y}|\text{domain})} \\ &= \frac{P(\mathbf{y}|\mathbf{x}, \text{domain})}{P(\mathbf{y}|\mathbf{x}_{\text{domain}})} \longleftarrow \text{We use “patterns”,} \\ &\quad \text{e.g., “Movie:”, “Review:”} \end{aligned}$$

# Noisy Channel

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \propto P(x|y)P(y)$$

P("It was great" | "A three-hour cinema master class.")  
P("It was terrible" | "A three-hour cinema master class.")



P("A three-hour cinema master class." | "It was great")  
P("A three-hour cinema master class." | "It was terrible")

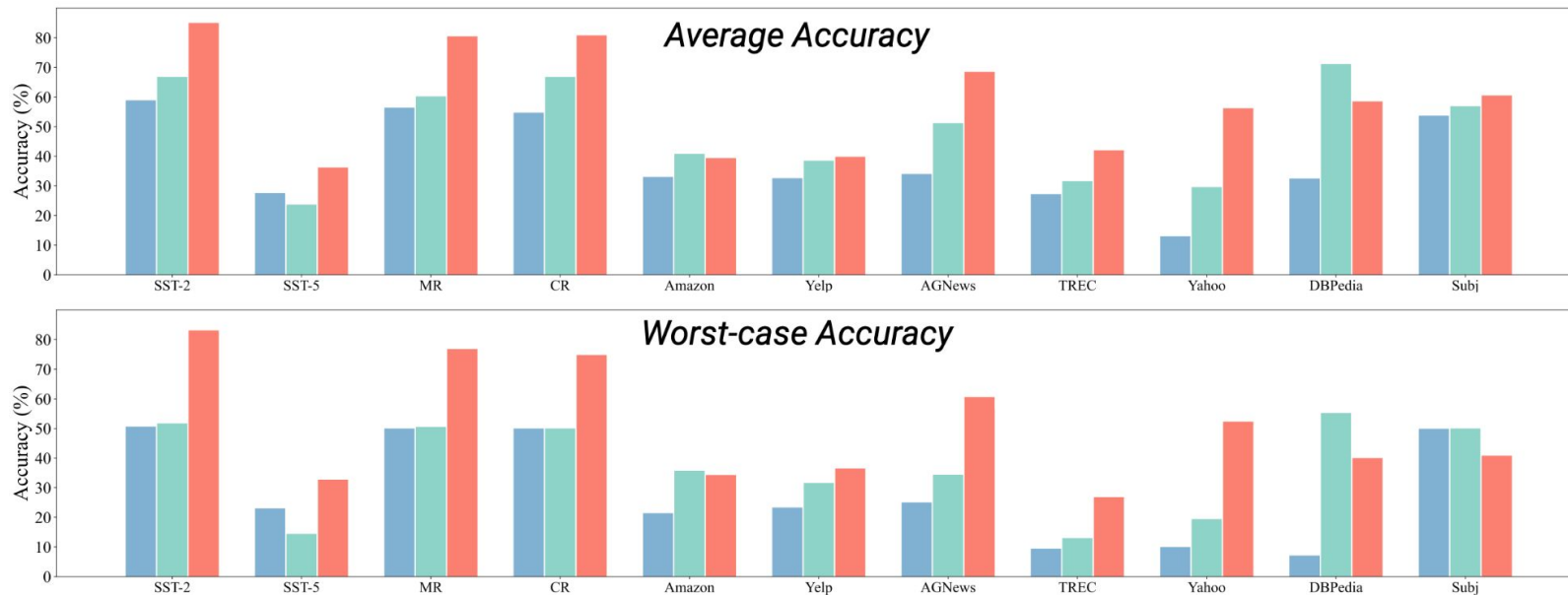
**Better calibration, More robust to distribution shift**

(between LM pretraining & prompting)

# Noisy Channel

(Original conditional prob) (+ calibration)

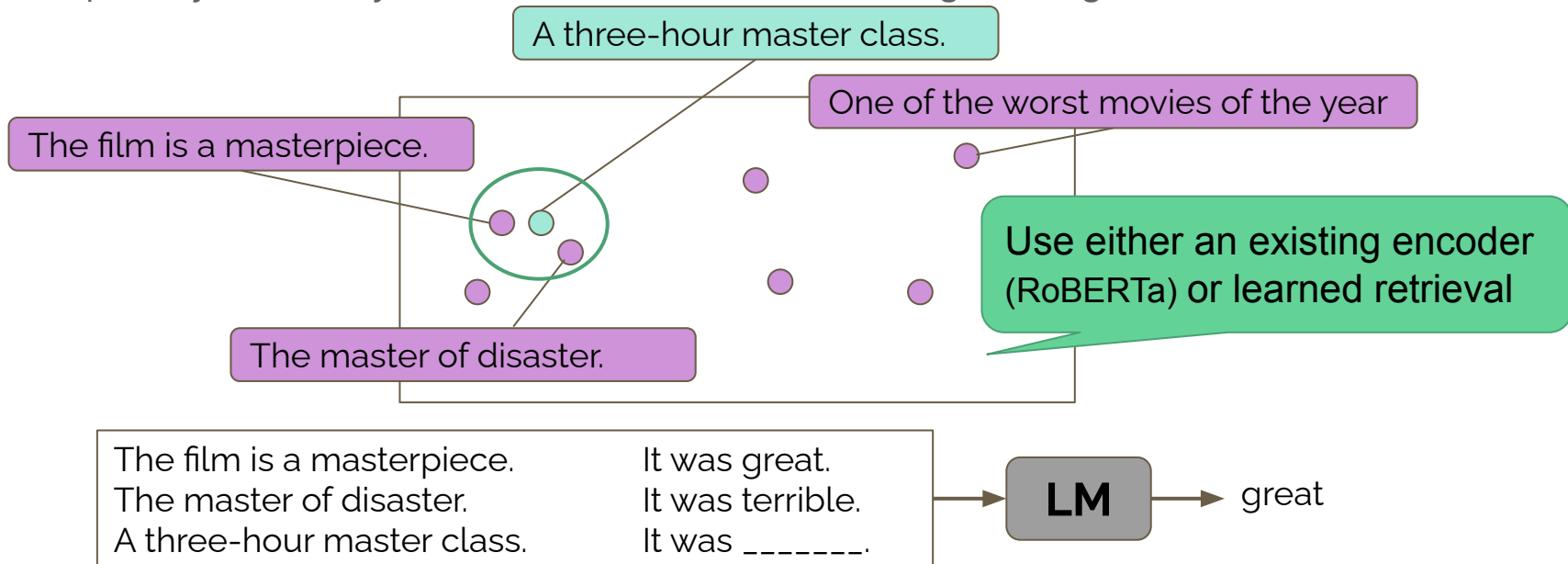
Direct Direct++ Channel



[Min et al. 2022](#). "Noisy Channel Language Model Prompting for Few-Shot Text Classification"

# How to choose the best $k$ examples?

Assumption: you already have the labeled data that is large enough



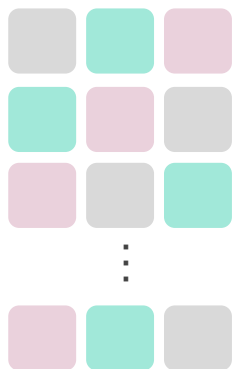
[Liu et al. 2021](#). What Makes Good In-Context Examples for GPT-3?  
[Rubin et al. 2021](#). “Learning To Retrieve Prompts for In-Context Learning”

# How to order k examples?

Review: A mostly tired reread  
of several other mob tales.  
Sentiment: negative

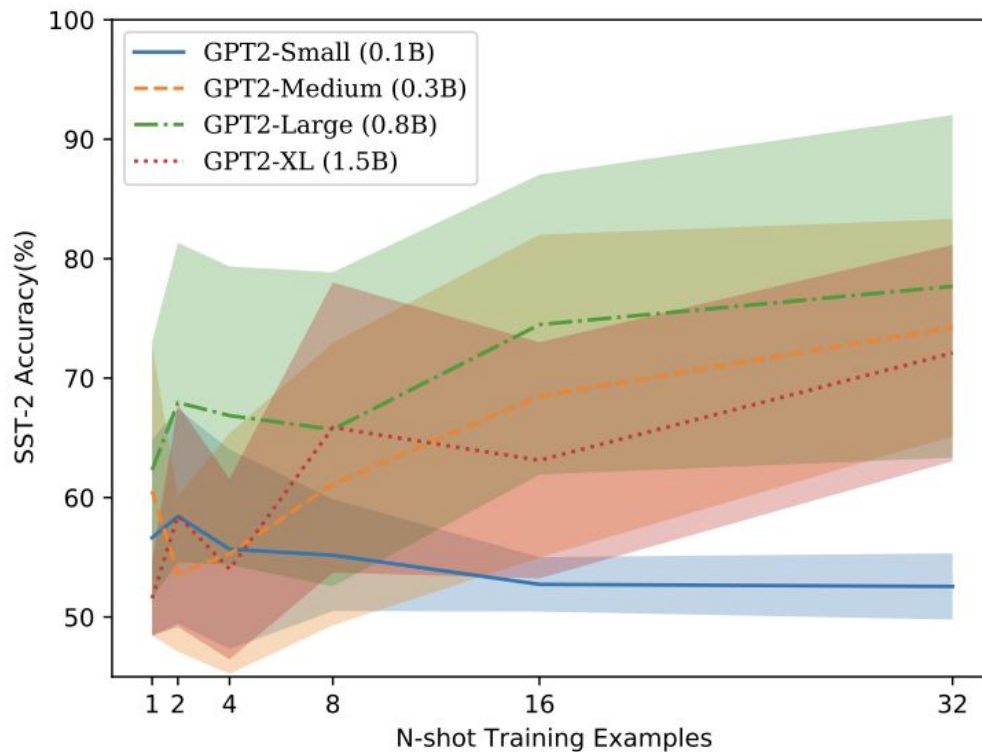
Review: The film is the  
masterpiece.  
Sentiment: negative

Review: One of the worst  
movies of the year.  
Sentiment: negative



Positive  
Negative  
Positive  
⋮  
Negative

# How to order k examples?





# How to order k examples?

Step 1: Generate  
**unlabeled dev set**

Step 2: **Score** each permutation  
based on unlabeled dev set

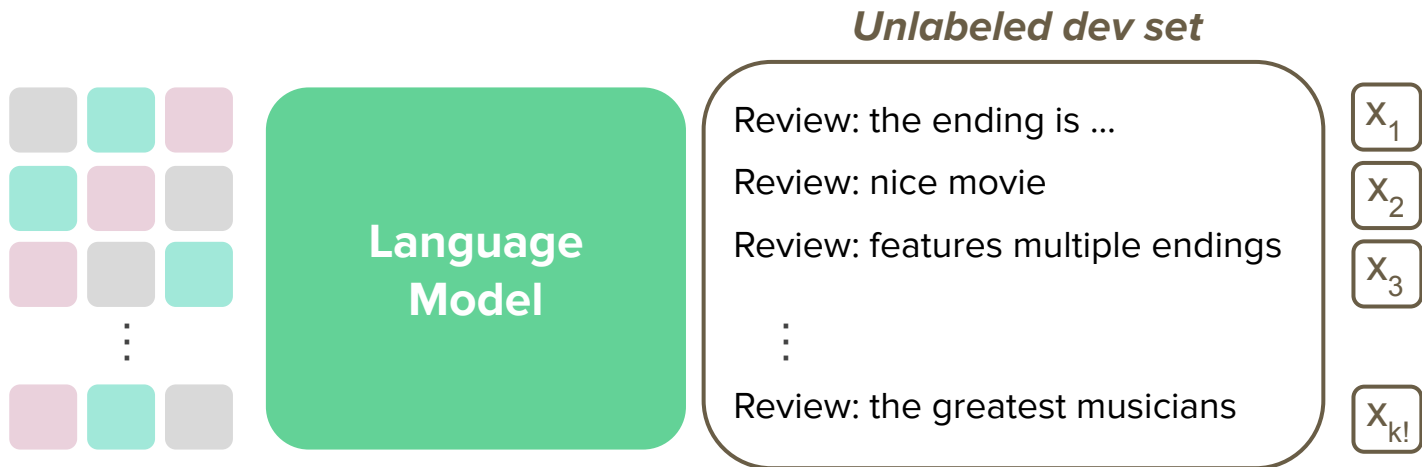
Step 3: **Choose** the best  
permutation!

# How to order k examples?

Step 1: Generate  
**unlabeled dev set**

Step 2: **Score** each permutation  
based on unlabeled dev set

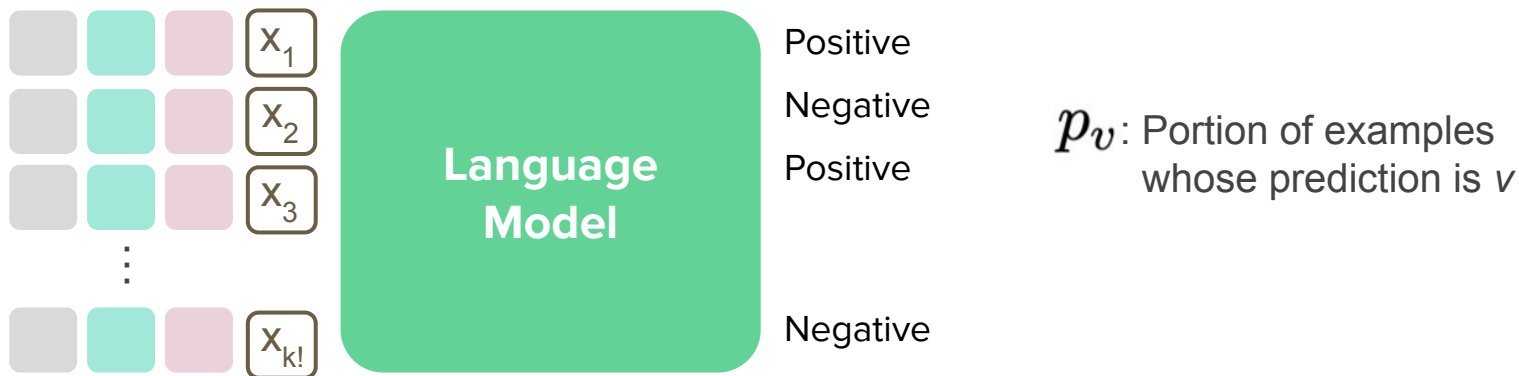
Step 3: **Choose** the best  
permutation!



# How to order k examples?

## 1. GlobalE

Intuition: model prediction over  $k!$  examples should be evenly distributed

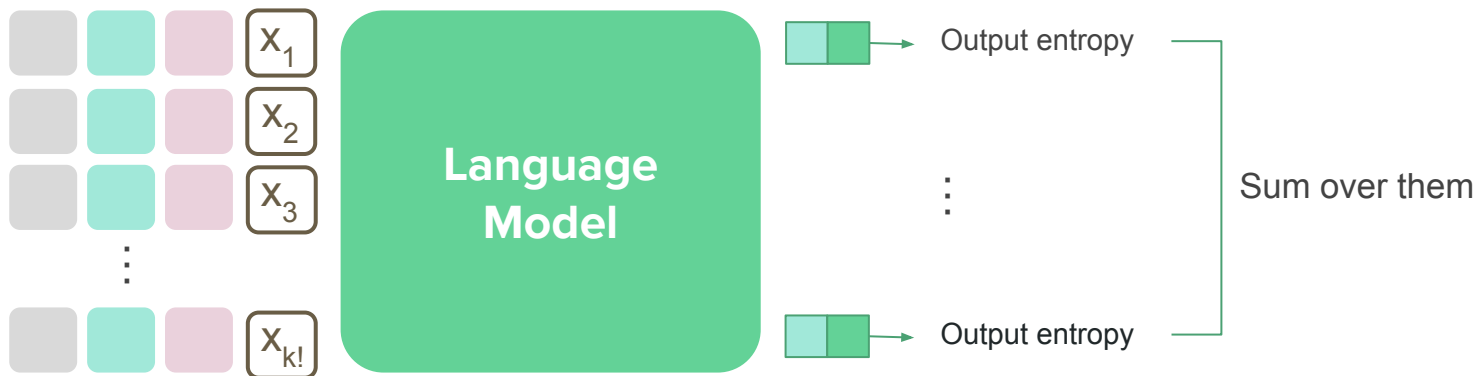


$$\text{GlobalE} = \sum_{v \in \mathcal{V}} (-p_v \log p_v)$$

# How to order k examples?

## 2. LocalE

Intuition: model output shouldn't be overly confident



$$\text{LocalE} = \sum_{1 \leq i \leq k} \sum_{v \in \mathcal{V}} (-P(v|x_i) \log P(v|x_i))$$

# Review

- LM prompting & In-context learning show promising results, but their performance is highly unstable/brittle
- Better scoring
  - Calibration
  - Domain Conditional PMI
  - Noisy Channel
  - (All for both zero-shot & in-context learning)
- Better formation of demonstrations
  - Better choice of demonstrative examples
  - Better ordering of demonstrative examples
  - (For in-context learning)

# True Few-Shot Learning (Perez et al., 2021)

*“We are unconsciously cheating on the data, and few-shot performance is overestimated”*

- Use of large development data
- Choice of patterns and verbalizers
- Choice of various hyperparameters



→ Later we will discuss more about **evaluation** (Part 5)!

# Transition: How/Why in-context learning works?

*Any arbitrary task*



**Language Model**

*A few-shot learner*



# Transition: How/Why in-context learning works?

*Any arbitrary task*



**Language Model**

*A few-shot learner*





# How/Why in-context learning works?

- Demonstrations do not teach a new task; instead, it is about locating an already-learned task during pretraining (Reynolds & McDonell, 2021)
- LMs do not exactly understand the meaning of their prompt (Webson & Pavlick, 2021)
- Demonstrations are about providing a latent concept so that LM generates coherent next tokens (Xie et al. 2022)
- In-context learning performance is highly correlated with term frequencies during pretraining (Razeghi et al. 2022)
- LMs do not need input-label mapping in demonstrations, instead, it uses the specification of the input & label distribution separately (Min et al. 2022)
- Data properties lead to the emergence of few-shot learning (burstiness, long-tailedness, many-to-one or one-to-many mappings, a Zipfian distribution) (Chan et al. 2022)

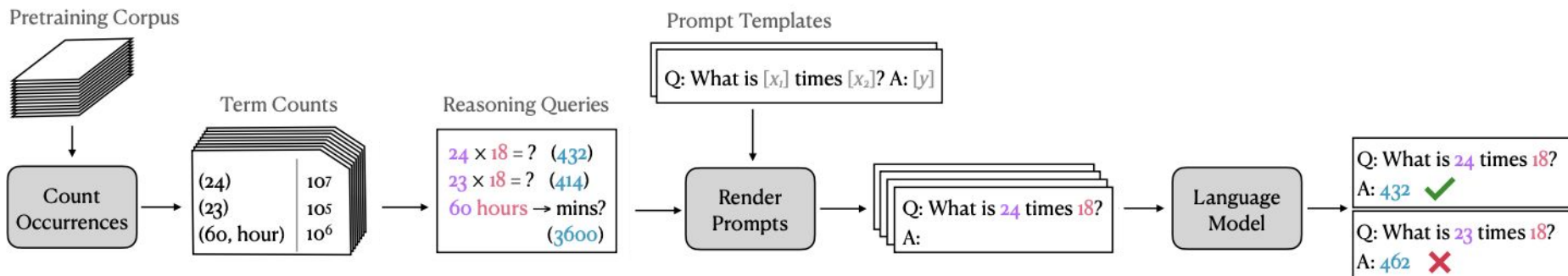
# How/Why in-context learning works?

- Demonstrations do not teach a new task; instead, it is about locating an already-learned task during pretraining (Reynolds & McDonell, 2021)
- LMs do not exactly understand the meaning of their prompt (Webson & Pavlick, 2021)
- Demonstrations are about providing a latent concept so that LM generates coherent next tokens (Xie et al. 2022)
- **In-context learning performance is highly correlated with term frequencies during pretraining** (Razeghi et al. 2022)
- **LMs do not need input-label mapping in demonstrations, instead, it uses the specification of the input & label distribution separately** (Min et al. 2022)
- Data properties lead to the emergence of few-shot learning (burstiness, long-tailedness, many-to-one or one-to-many mappings, a Zipfian distribution) (Chan et al. 2022)

# Impact of Pretraining Term Frequencies

**In-context learning performance is highly correlated with term frequencies during pretraining**

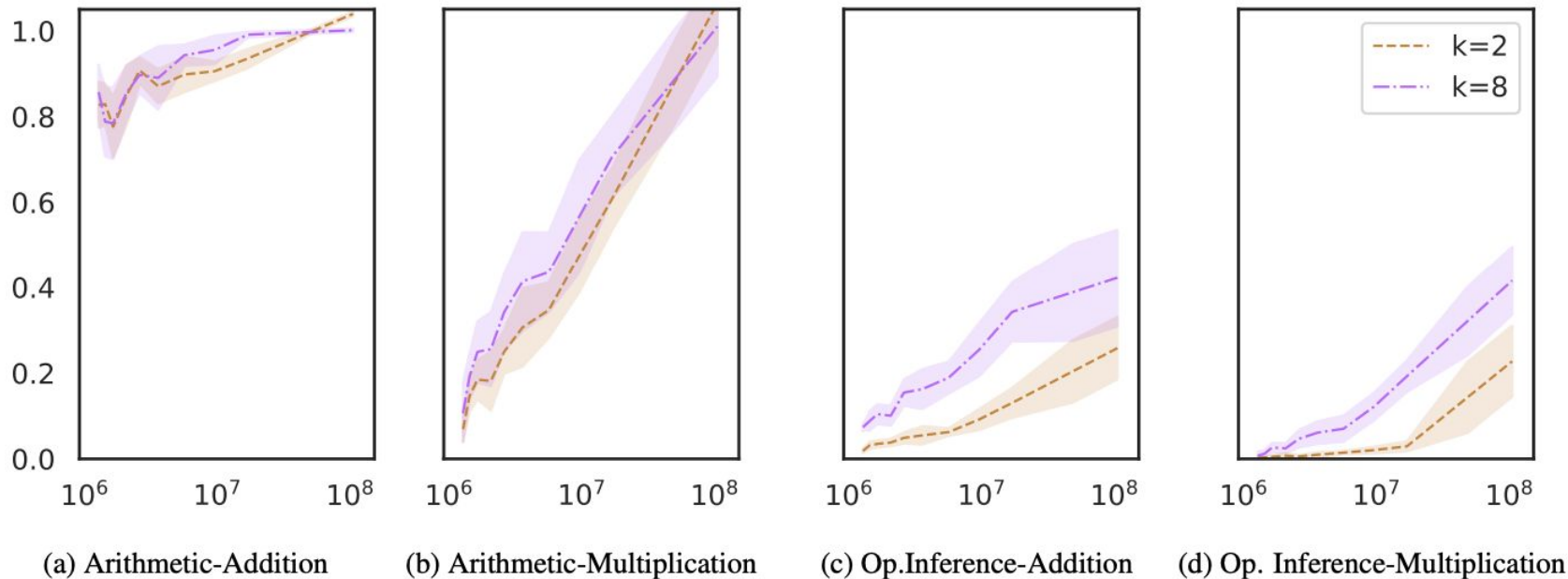
- For each task, identify relevant terms from each instance—numbers and units
- Count co-occurrences of these terms in the pretraining data (term pairs or triples within a fixed window)



[Razeghi et al. 2022](#). "Impact of Pretraining Term Frequencies on Few-Shot Reasoning"

# Impact of Pretraining Term Frequencies

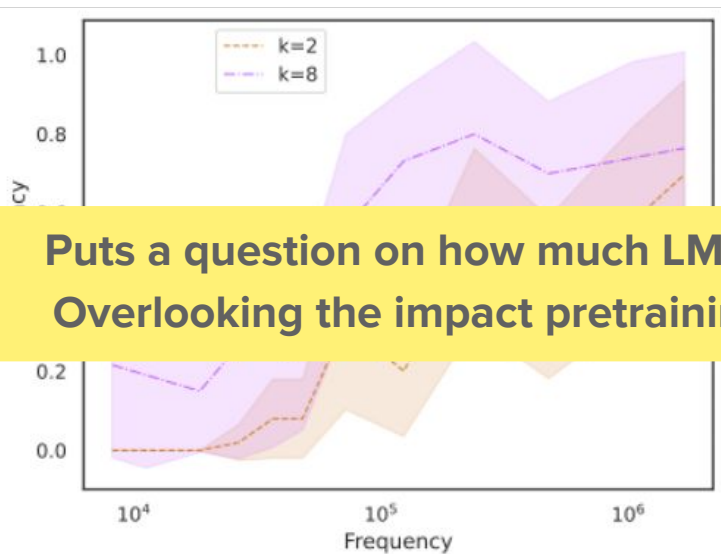
In-context learning performance is highly correlated with term frequencies during pretraining



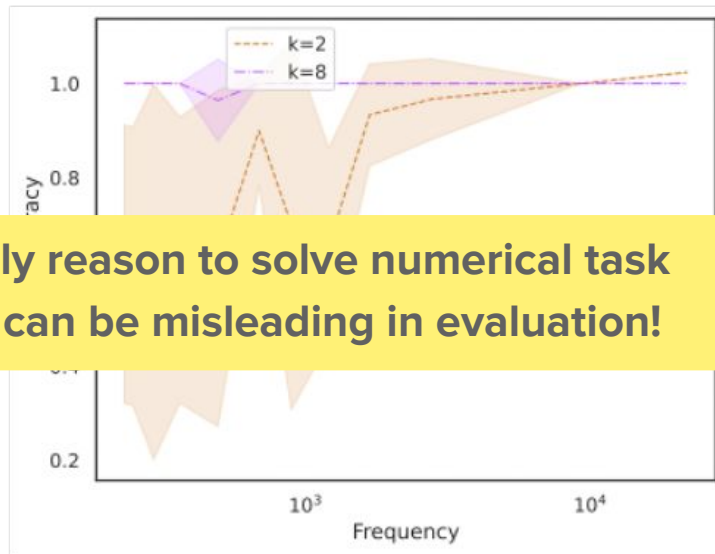
# Impact of Pretraining Term Frequencies

In-context learning performance is highly correlated with term frequencies during pretraining

Time-Unit Conversion Year to Month



Time-Unit Conversion Decade to Year



Puts a question on how much LMs actually reason to solve numerical task  
Overlooking the impact pretraining data can be misleading in evaluation!

# Impact of input-label mapping

## In-context learning does not necessitate correct input-label mapping

**Input:** An effortlessly accomplished and richly resonant work.

**Label:** positive

**Input:** A mostly tired retread of several other mob tales.

**Label:** negative

**Input:** A three-hour master class.

**Label:** \_\_\_\_\_

Language  
Model

**Input:** An effortlessly accomplished and richly resonant work.

**Label:** **negative**

**Input:** A mostly tired retread of several other mob tales.

**Label:** **positive**

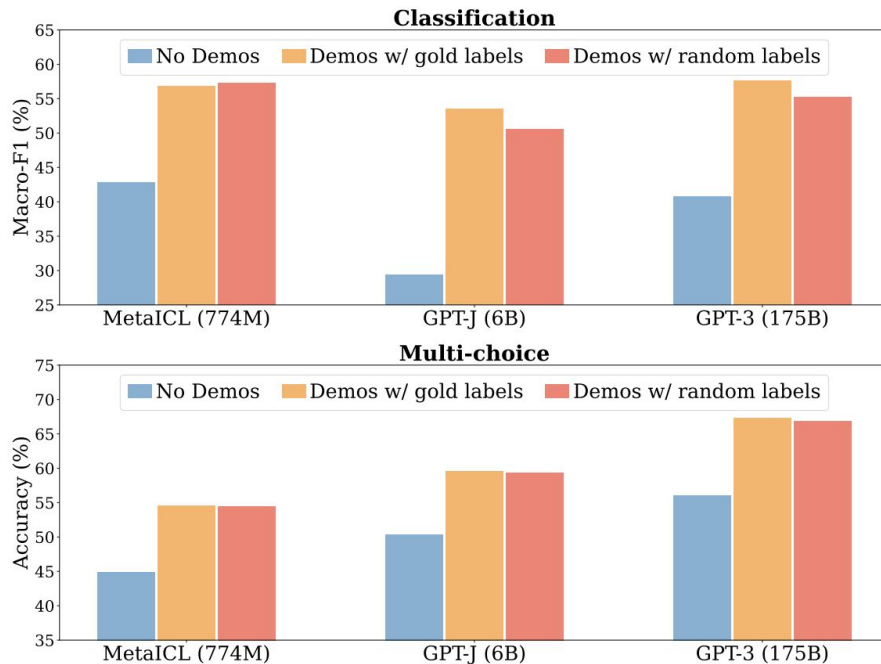
**Input:** A three-hour master class.

**Label:** \_\_\_\_\_

Language  
Model

# Impact of input-label mapping

In-context learning does not necessitate correct input-label mapping



[Min et al. 2022](#). "Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?"

# Impact of input-label mapping

In-context learning does not necessitate correct input-label mapping

**Input:** An effortlessly accomplished and richly resonant work.

**Label:** positive

**Input:** A mostly tired retread of several other mob tales.

**Label:** negative

**Input:** A three-hour master class.

**Label:** \_\_\_\_\_

Language  
Model



# Impact of input-label mapping

In-context learning does not necessitate correct input-label mapping

**Input:** Colour-printed lithograph. Very good condition.

**Label:** positive

**Input:** Many accompanying marketing ... meaning.

**Label:** negative

**Input:** A three-hour master class.

**Label:** \_\_\_\_\_

Language  
Model

Removing correct **input distribution**  
significantly drops performance

# Impact of input-label mapping

In-context learning does not necessitate correct input-label mapping

**Input:** An effortlessly accomplished and richly resonant work.

**Label:** Unanimity

**Input:** A mostly tired retread of several other mob tales.

**Label:** Wave

**Input:** A three-hour master class.

**Label:** \_\_\_\_\_

Language  
Model

Removing correct **input distribution**  
significantly drops performance

Removing correct **label space**  
significantly drops performance

Input and label distributions matter *independently*

# Summary & Open questions

- In-context learning has been a promising few-shot learning approach
  - No need for gradient updates → Much easier to use large models!  
(Even compared to parameter-efficient tuning covered in Section 3)
- Better calibration, better scoring of model outputs, better formation of demonstrations lead to great improvements
  - How to make it less sensitive?
  - It increases inference cost – how to make it efficient?
  - How to scale it (longer context, more training examples, wider range of tasks)?
- Need to be cautious in evaluation
- Still in progress on understanding how/why it works, with papers showing that in-context learning is about *task location* rather than learning a *new* task
  - Can we predict whether in-context learning would work on a given task or not?

# Future Work and Open Questions

## Better in-context-learning

- Make it less sensitive to the order of training examples or patterns/verbalizers?
- It increases inference cost – how to make it efficient?
- How to scale it (longer context, more training examples)?

## Better understanding

- Can we better understand how and why it works?
- Can we predict whether in-context learning would work on a given task or not?

# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation benchmark [Arman]
- 17:05–17:25 Part 6: Meta-training [Arman]
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]

# Part 3: Gradient-based LM task adaptation

---

Slides available at:

<https://github.com/allenai/acl2022-zerofewshot-tutorial>

# In this section...

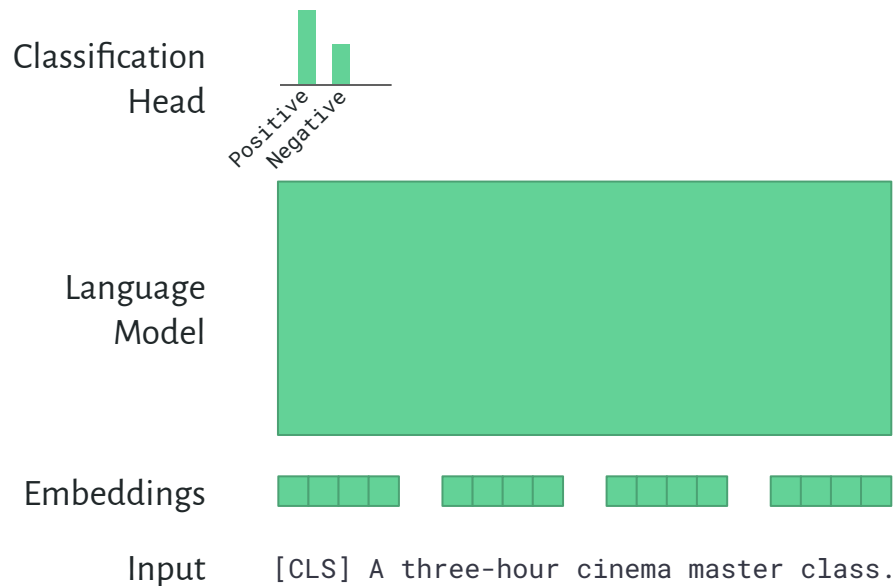
We cover few-shot learning approaches that use gradient information.

- Methods that finetune all of the model weights:
  - Traditional finetuning, prompt-based finetuning, and PET
  - Sensitivity analysis
- Parameter-efficient finetuning methods:
  - Input only: prompt search, prompt tuning
  - Internal updates: BitFit, Adapter, Compacter, LoRa, (IA)<sup>3</sup>

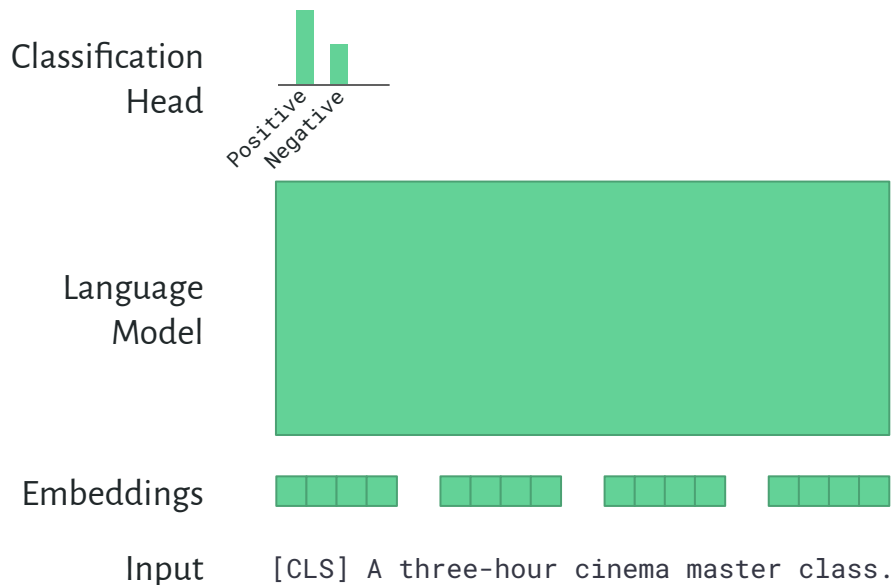
# Full Finetuning Approaches



# Traditional finetuning



# Traditional finetuning



Default finetuning recommendations are unstable in few-shot settings.

Stability can be improved by:

- Using smaller learning rates
- Training for more iterations
- Re-including debiasing terms in ADAM optimizer

However finetuning still underperforms other methods.

[Dodge et al., 2020](#). "Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping"

[Mosbach et al., 2020](#). "On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines"

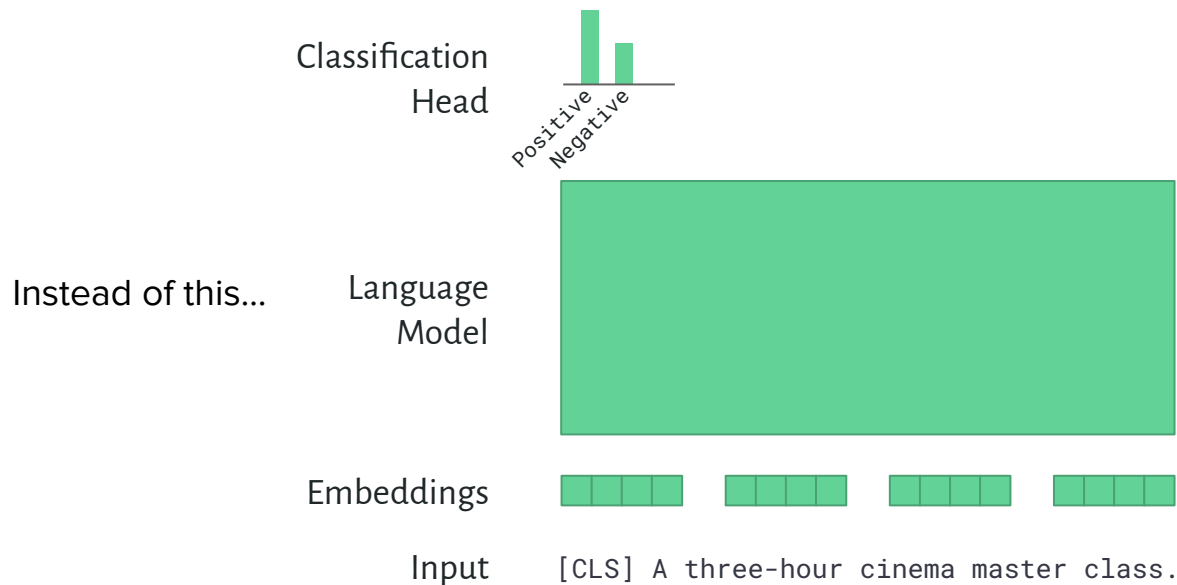
[Zhang et al., 2020](#). "Revisiting Few-sample BERT Fine-tuning"

# Prompt-based finetuning

Combine prompting with finetuning.

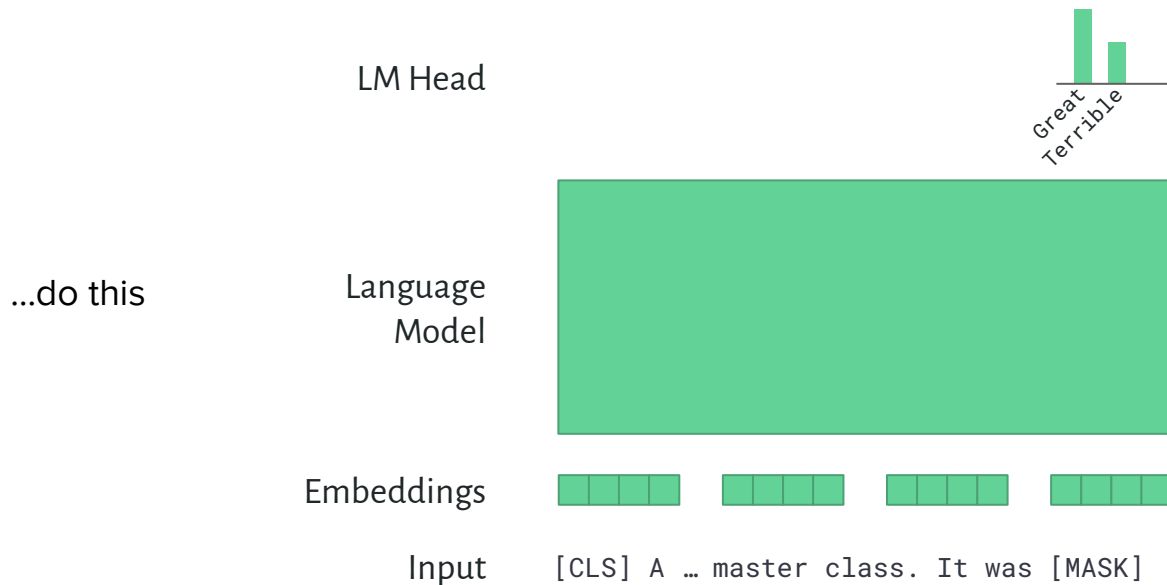
# Prompt-based finetuning

Combine prompting with finetuning.



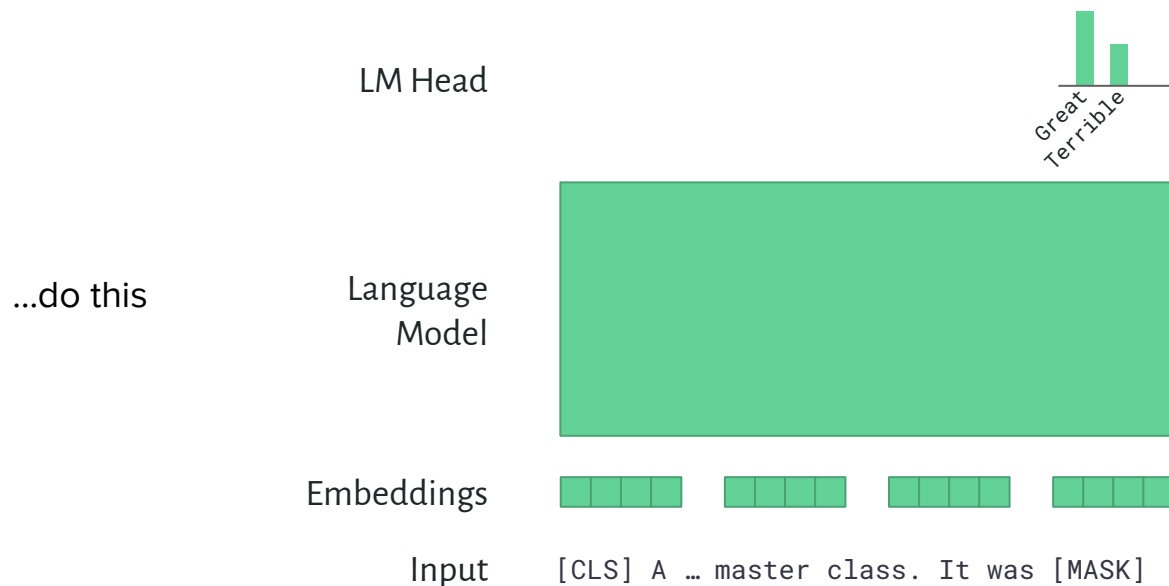
# Prompt-based finetuning

Combine prompting with finetuning.



# Prompt-based finetuning

Combine prompting with finetuning.



**T5** (enc-dec):  
[Roberts et al., 2019](#)

**PET** (mlm):  
[Schick and Schutze, 2020a](#)  
[Schick and Schutze, 2020b](#)

# Prompt-based finetuning

Loss objectives for masked language models:

- PET (Schick and Schütze, 2020) uses cross-entropy over the set of label tokens.

$$q(y|x) = \frac{\exp(\llbracket G_m(x) \rrbracket_y)}{\sum_{y' \in \mathcal{Y}} \exp(\llbracket G_m(x) \rrbracket_{y'})} \quad \mathcal{L} = \text{CE}(q(y^*|x), y^*)$$

- ADAPET (Tam et al., 2021) uses binary cross entropy of the LM probabilities.

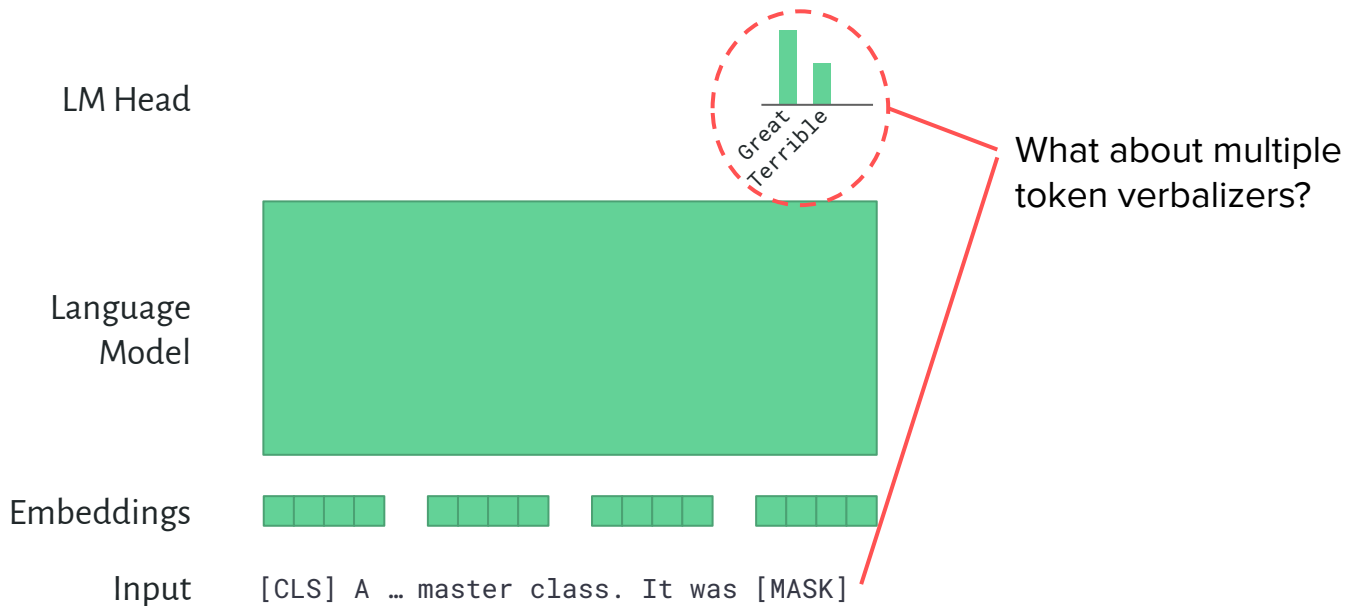
$$q(y|x) = \frac{\exp(\llbracket G_m(x) \rrbracket_y)}{\sum_{v' \in \mathcal{V}} \exp(\llbracket G_m(x) \rrbracket_{v'})} \quad \mathcal{L}_D = \text{BCE}(q(y^*|x), 1) + \sum_{y \neq y^*} \text{BCE}(q(y|x), 0)$$

[Schick and Schütze, 2020](#). "Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference"

[Tam et al., 2021](#). "Improving and Simplifying Pattern Exploiting Training"

# Prompt-based finetuning

Combine prompting with finetuning.





# Prompt-based finetuning

How to address multi-token verbalizers for masked language models:

# Prompt-based finetuning

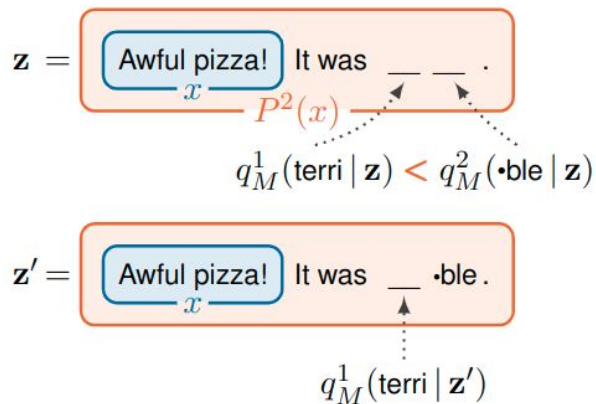
How to address multi-token verbalizers for masked language models:

1. Don't use them.

# Prompt-based finetuning

How to address multi-token verbalizers for masked language models:

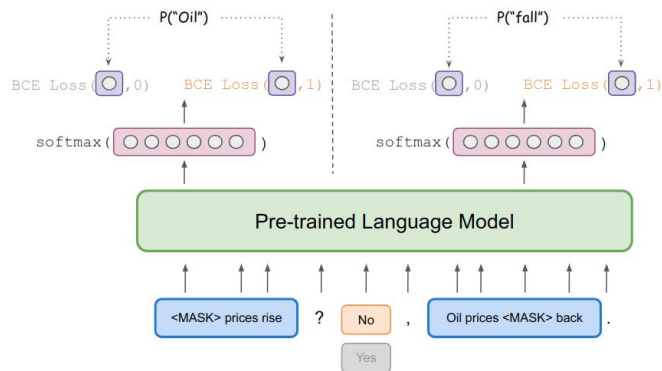
1. Don't use them.
2. Hinge loss (training) + Autoregressive decoding (inference)



# Prompt-based finetuning

How to address multi-token verbalizers for masked language models:

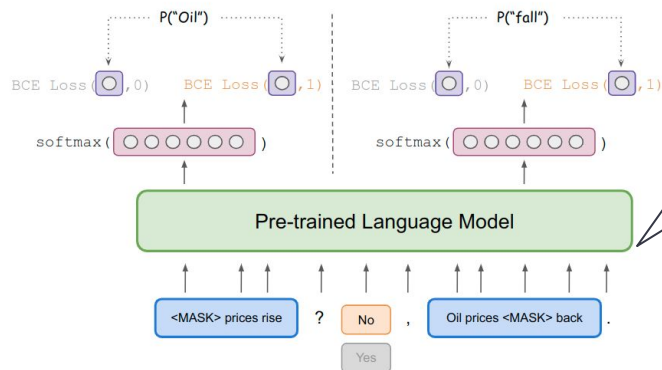
1. Don't use them.
2. Hinge loss (training) + Autoregressive decoding (inference)
3. Label conditioning, i.e., noisy channel approach



# Prompt-based finetuning

How to address multi-token verbalizers for masked language models:

1. Don't use them.
2. Hinge loss (training) + Autoregressive decoding (inference)
3. Label conditioning, i.e., noisy channel approach




Performance heavily dependent on masking rate, which is hard to determine in "true" few shot settings.

- [Perez et al., 2021](#)

# Prompt-based finetuning vs. traditional finetuning

Prompt-based finetuning has higher accuracy than traditional finetuning.

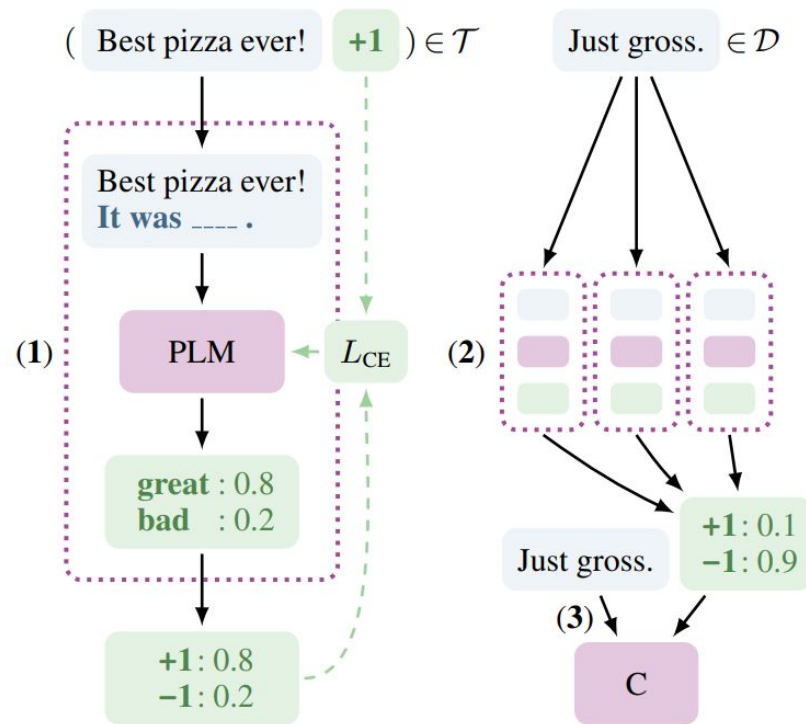
- “On average, prompt-based finetuning outperforms traditional finetuning by up to 30% on classification tasks” [Lester et al., 2021](#)
- Up to 30% improvement in accuracy on classification tasks [Gao et al., 2020](#)
- Outperforms traditional finetuning on classification tasks [Lester et al., 2021](#)



We can make  
prompt-based  
finetuning even  
better!

# Pattern exploiting training (PET)

1. Train an ensemble of classifiers using prompt-based finetuning in few-shot setting.
2. Collect weak labels for a pool of unlabeled data.
3. Use to train final classifier w/ traditional finetuning.

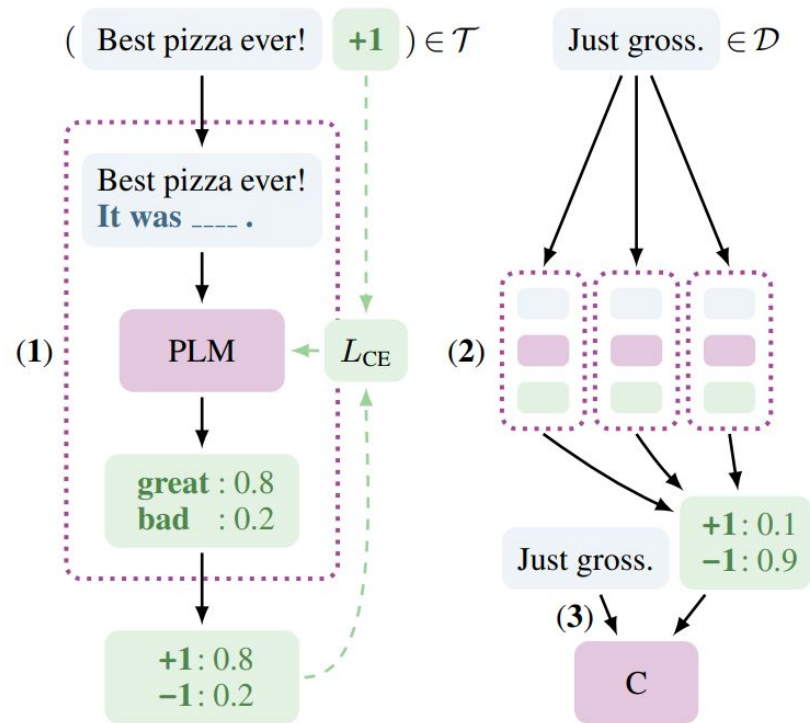


# Pattern exploiting training (PET)

## Iterative PET (iPET)

Train several generations of PET models on datasets of increasing size.

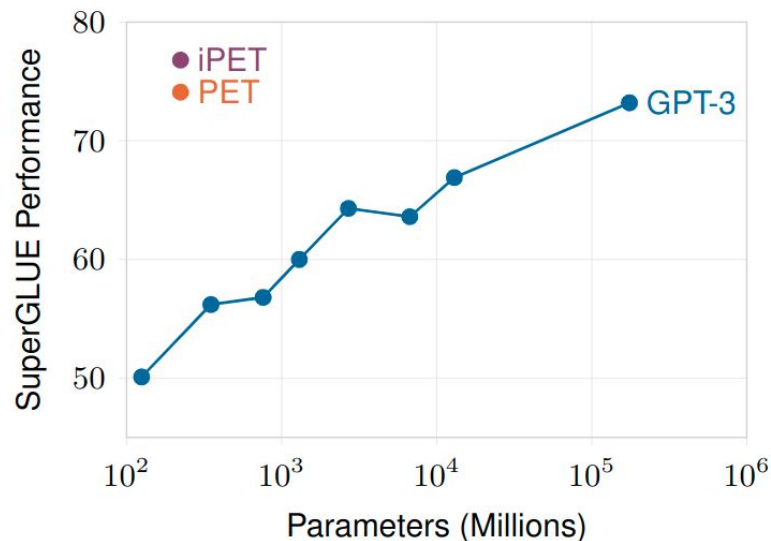
- Use output of other models to obtain labels.
- Select examples:
  - That models are more confident on.
  - That maintain label balance.





# PET - Results

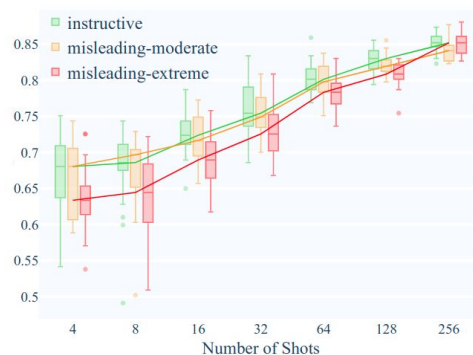
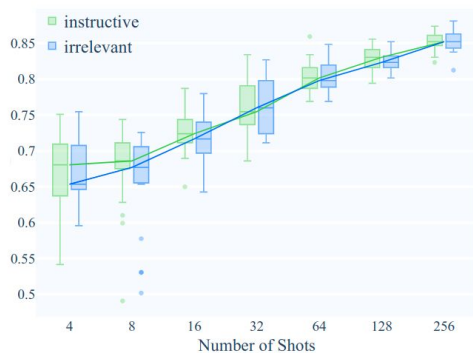
- PET outperforms GPT-3 while using 1000x less parameters.
- Distillation approach consistently improves prompt-based finetuning.



# Sensitivity to prompt

“Null” prompts that just concatenate inputs with a mask token often perform competitively with manually written prompts. [Logan et al., 2021](#)

There is often no statistically significant difference in accuracy between “instructive” and “irrelevant” prompts. [Webson and Pavlick, 2021](#)



[Webson and Pavlick, 2021](#). "Do Prompt-Based Models Really Understand the Meaning of Their Prompts?"

# Sensitivity to prompt

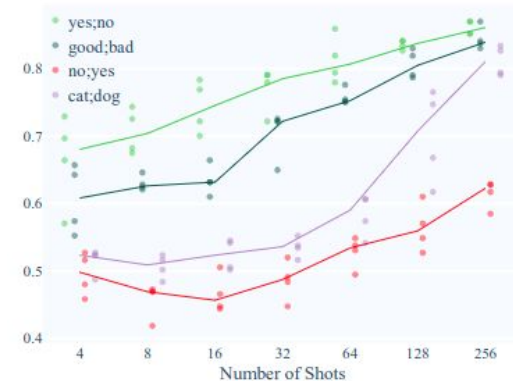
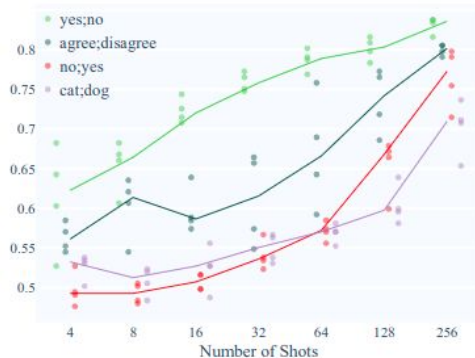
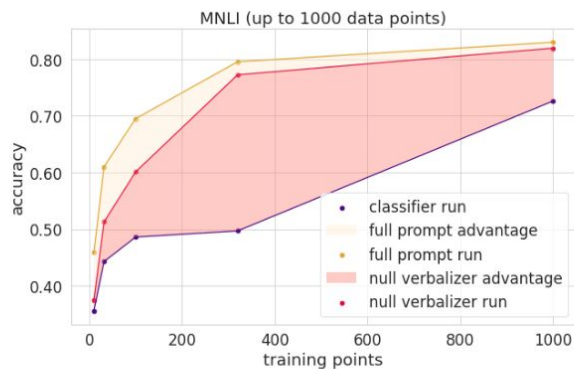
“Null” prompts that just concatenate inputs with a mask token often perform competitively with manually written prompts. [Logan et al., 2021](#)

There is often no statistically significant difference in accuracy between “instructive” and “irrelevant”/“misleading” prompts. [Webson and Pavlick, 2021](#)

Task-related prompts typically rank higher than null and generic prompts in terms of average accuracy. [Schick and Schütze, 2021](#)

# Sensitivity to Verbalizer

Replacing verbalizers with random tokens from the vocabulary and flipping verbalizers can substantially decrease performance.



[Le Scao and Rush, 2021](#). "How Many Data Points is a Prompt Worth?"

[Webson and Pavlick, 2021](#). "Do Prompt-Based Models Really Understand the Meaning of Their Prompts?"

# Summary of prompt-based finetuning

Prompt-based finetuning methods are competitive with or outperform in-context learning in few-shot settings, especially PET.

However, there are some tradeoffs

	Model Size	Task-Specific Parameters
In-Context Learning	10B - 100B	Effectively None
Prompt-Based Finetuning	100M - 1B	All

# Summary of prompt-based finetuning

Prompt-based finetuning methods are competitive with or outperform in-context learning in few-shot settings, especially PET.

However, there are some tradeoffs

	Model Size	Task-Specific Parameters
In-Context Learning	10B - 100B	Effectively None
Prompt-Based Finetuning	100M - 1B	<b>All</b>

# Parameter-Efficient Finetuning

# Parameter-Efficient Finetuning

	Model Size	Task-Specific Parameters
In-Context Learning	10B - 100B	Effectively None
Prompt-Based Finetuning	100M - 1B	All
Parameter-Eff. Finetuning	100M - 1B	<1% of model parameters



# Parameter-Efficient Finetuning

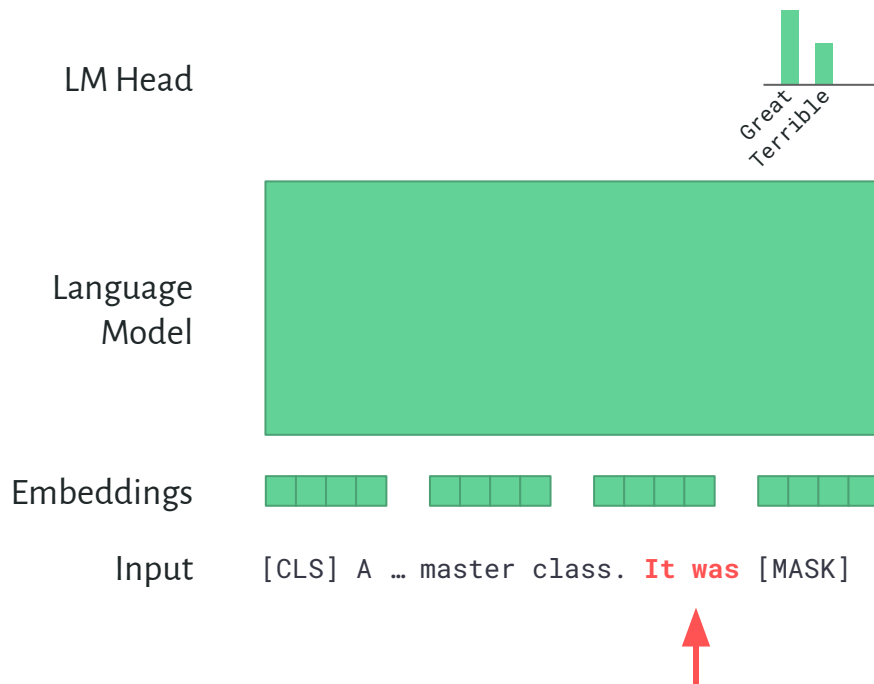
	Model Size	Task-Specific Parameters
In-Context Learning	10B - 100B	Effectively None
Prompt-Based Finetuning	100M - 1B	All
Parameter-Eff. Finetuning	100M - 1B	<1% of model parameters

Methods described in order of increasing competitiveness with prompt-based finetuning.

# Input-level modifications

Two types:

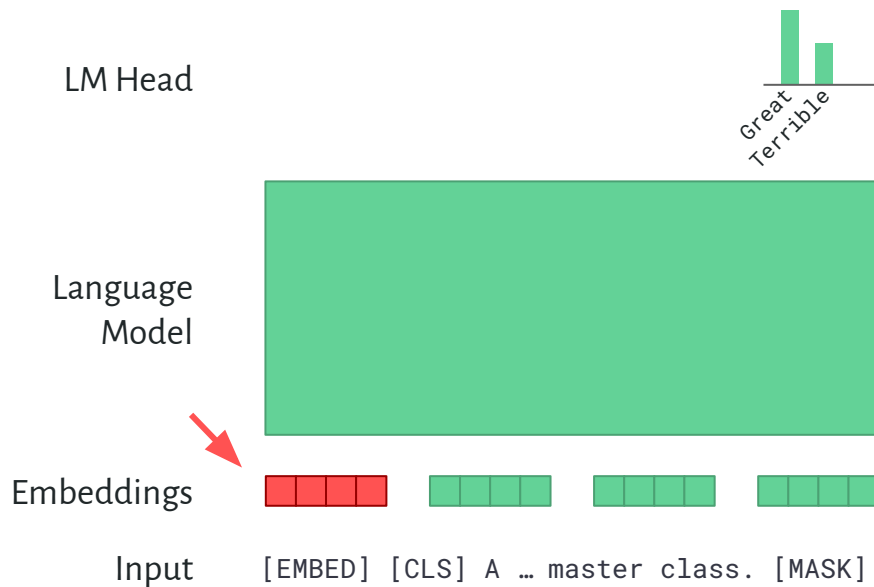
1. **Prompt search** methods try to learn the tokens in the prompt.



# Input-level modifications

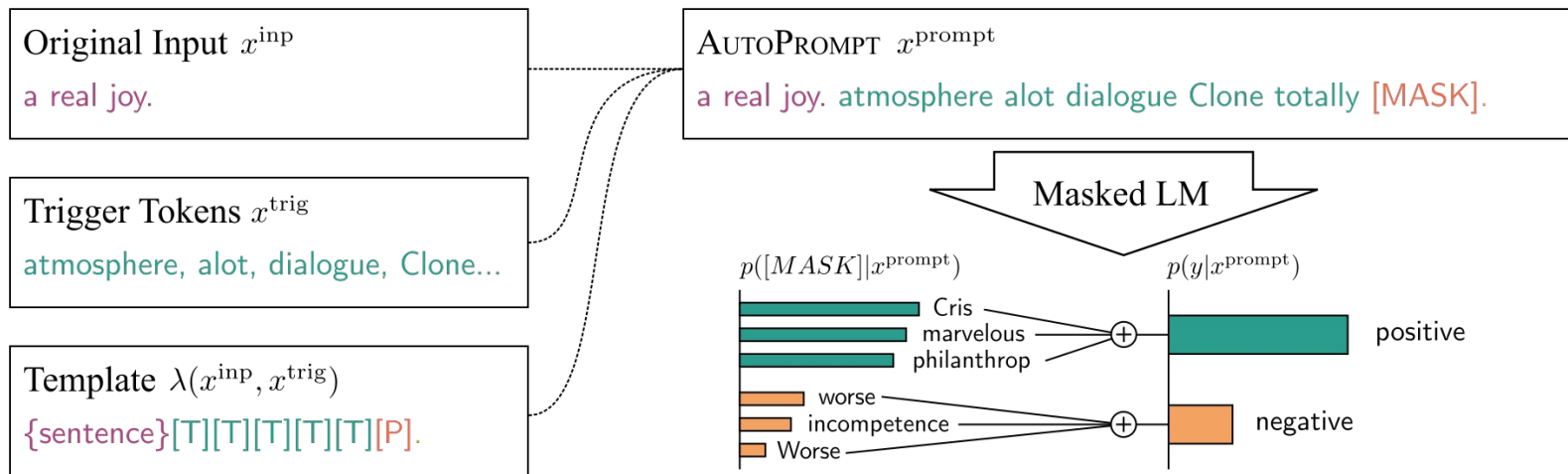
Two types:

1. Prompt search methods try to learn the tokens in the prompt.
2. **Prompt tuning** methods introduce novel embeddings that are learned using gradient descent.



# Prompt Search Methods

**AutoPrompt:** Iteratively updates tokens in the pattern using a gradient-guided search. ([Shin et al. 2020](#))

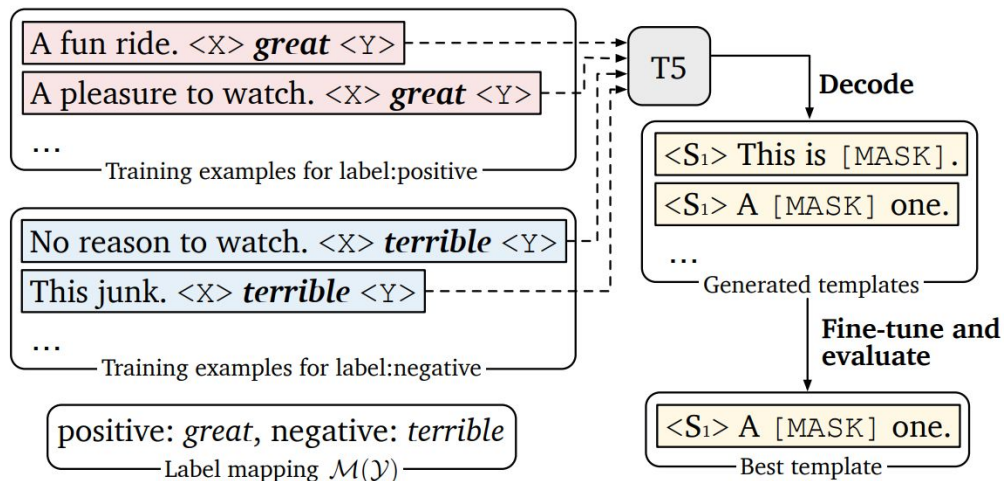


# Prompt Search Methods

**AutoPrompt:** Iteratively updates tokens in the pattern using a gradient-guided search. ([Shin et al. 2020](#))

**LM-BFF:** Uses T5's span filling capabilities to decode patterns all-at-once.

([Gao et al. 2020](#))



# Prompt Search Methods

**AutoPrompt:** Iteratively updates tokens in the pattern using a gradient-guided search. ([Shin et al. 2020](#))

**LM-BFF:** Uses T5's span filling capabilities to decode patterns all-at-once. ([Gao et al. 2020](#))

Performance-wise, AutoPrompt has been shown to significantly underperform relative to prompt-based finetuning. ([Logan et al., 2021](#))

# Prompt Search Methods

**AutoPrompt:** Iteratively updates tokens in the pattern using a gradient-guided search. ([Shin et al. 2020](#))

**LM-BFF:** Uses T5's span filling capabilities to decode patterns all-at-once. ([Gao et al. 2020](#))

Performance-wise, AutoPrompt has been shown to significantly underperform relative to prompt-based finetuning. ([Logan et al., 2021](#))

LM-BFF results only show that generated prompts are competitive with manually written prompts in prompt-based finetuning setting.

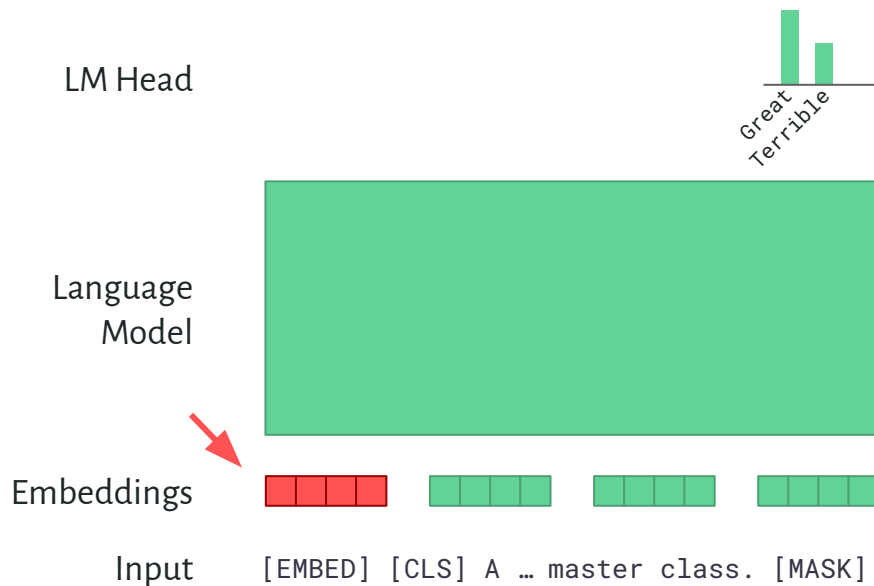
# Prompt Tuning

Learn embeddings for placeholder tokens in the pattern.

Aliases:

- WARP ([Hambarzumyan et al., 2021](#))
- OptiPrompt ([Zhong et al., 2021](#))
- Prompt Tuning ([Lester et al., 2021](#))
- P-Tuning\* ([Liu et al., 2021](#))

\*Encodes embeddings with an LSTM before feeding to model.





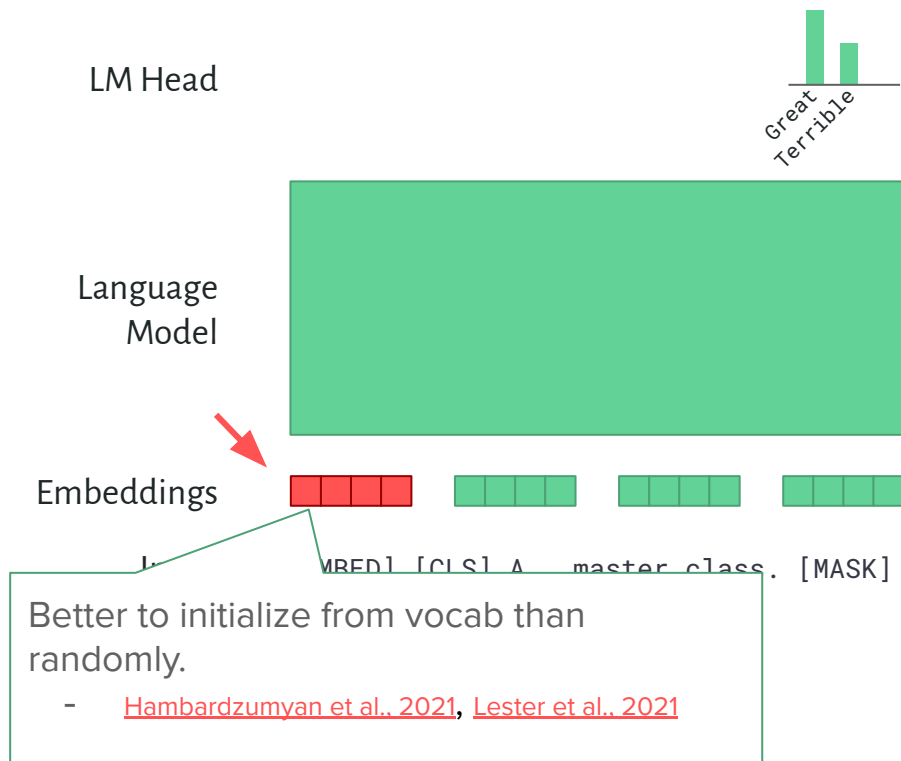
# Prompt Tuning

Learn embeddings for placeholder tokens in the pattern.

Aliases:

- WARP ([Hambarzumyan et al., 2021](#))
- OptiPrompt ([Zhong et al., 2021](#))
- Prompt Tuning ([Lester et al., 2021](#))
- P-Tuning\* ([Liu et al., 2021](#))

\*Encodes embeddings with an LSTM before feeding to model.



# Prompt Tuning

While prompt tuning can be competitive with full model tuning in full data settings, results are mixed in few-shot settings.

- [Hambarzumyan et al. \(2021\)](#) find prompt tuning performs between PET and iPET on CB and RTE, when initialized from manually written prompts.
- [Logan et al. \(2021\)](#) and [Liu et al. \(2022\)](#) find that prompt tuning consistently performs worse than full finetuning methods on 20 classification tasks, with up to ~30% drops in absolute accuracy.

# Prompt Tuning

While prompt tuning can be competitive with model tuning in full data settings, results are mixed in few-shot settings.

- [Hambard](#) compares performance between PET and iPET on CB and iCB tasks using different prompts.
- [Logan](#) compares performance between PET and iPET on 20 classification tasks, with up to ~30% improvement in some cases.

**Stay tuned for PPT  
in meta-training  
section!**

# Summary of input-level modifications

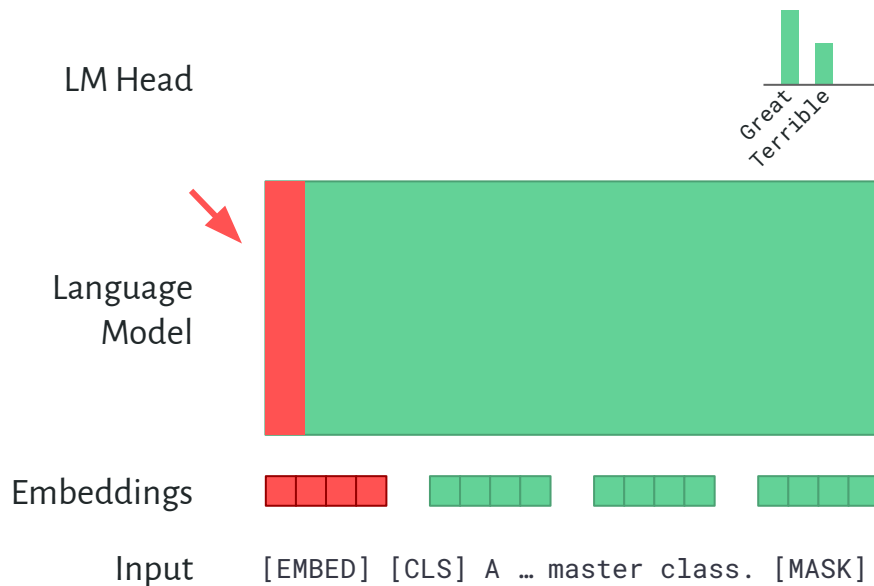
	Pattern	LM	Few-Shot Experiments (Prompt Only)	Few-Shot Experiments (Full Finetuning)
AutoPrompt	Discrete	*BERT	✓	✗
LM-BFF	Discrete	*BERT	✗	✓
WARP	Continuous	*BERT	✓	✗
P-Tuning	Continuous	GPT-2 / *BERT	✗	✓
OptiPrompt	Continuous	*BERT	✗	✗
Prompt Tuning	Continuous	T5	✗	✗

# Prefix Tuning / Soft Prompts

Idea: Learn contextualized embeddings as well as input embeddings.

Prefix Tuning does this using projection matrices.

Soft Prompts does this using additive perturbations.



[Li and Liang, 2021](#). “Prefix-Tuning: Optimizing Continuous Prompts for Generation”

[Qin and Eisner, 2021](#). “Learning How to Ask: Querying LMs with Mixtures of Soft Prompts”

# Prefix Tuning / Soft Prompts

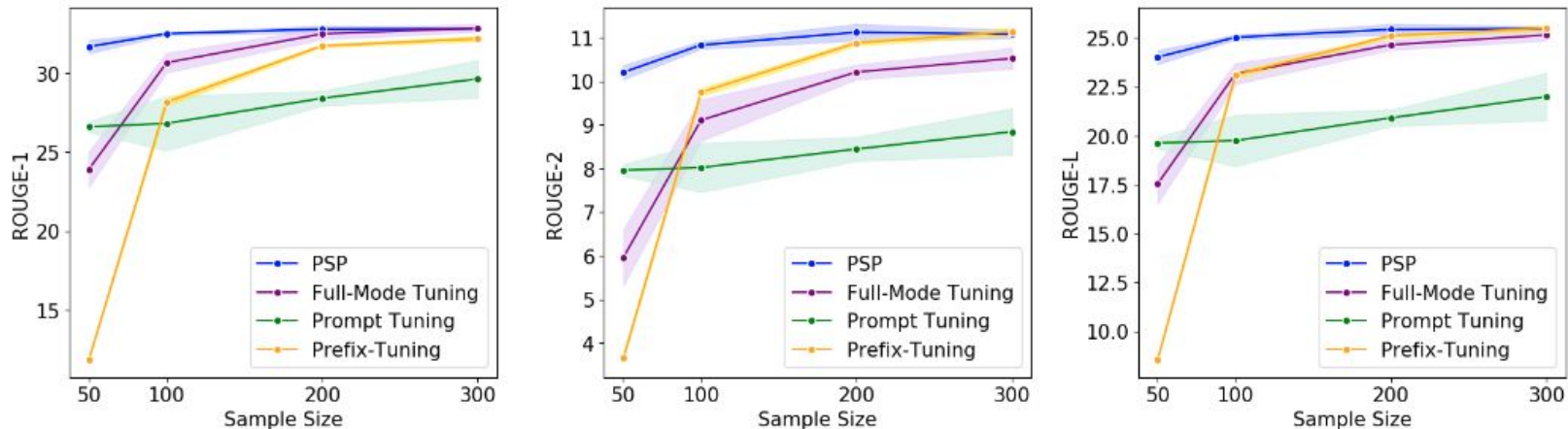
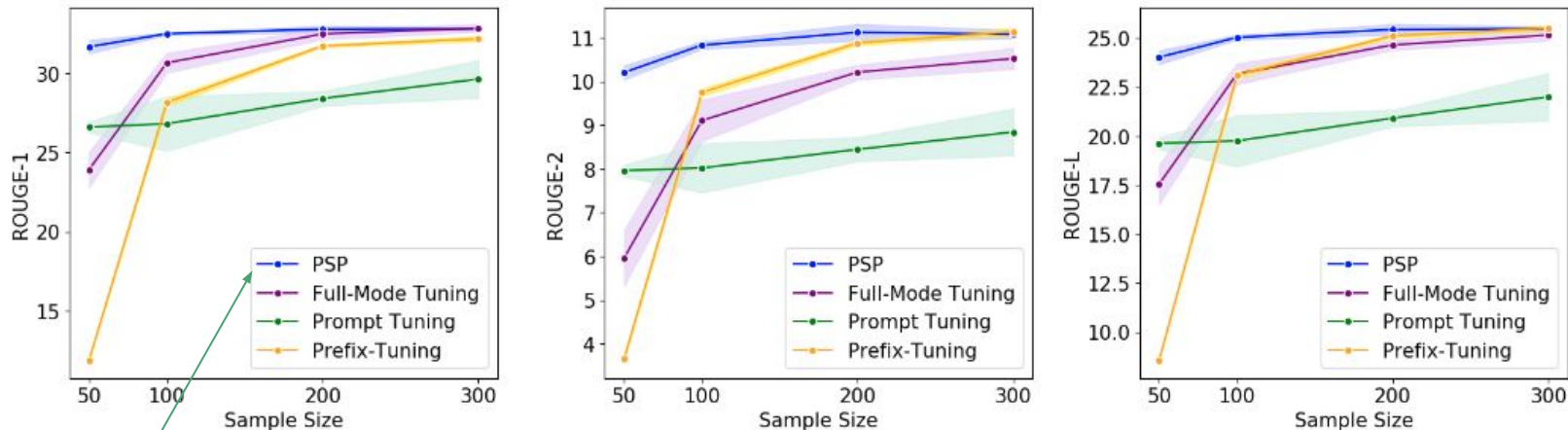


Figure 5:  $k$ -shot summarization results on XSum.

# Prefix Tuning / Soft Prompts



Meta-Training

Figure 5:  $k$ -shot summarization results on XSum.

# BitFit

BitFit tunes only the bias terms in self-attention and MLP layers.

$$\mathbf{Q}^{m,\ell}(\mathbf{x}) = \mathbf{W}_q^{m,\ell} \mathbf{x} + \mathbf{b}_q^{m,\ell}$$

$$\mathbf{K}^{m,\ell}(\mathbf{x}) = \mathbf{W}_k^{m,\ell} \mathbf{x} + \mathbf{b}_k^{m,\ell}$$

$$\mathbf{V}^{m,\ell}(\mathbf{x}) = \mathbf{W}_v^{m,\ell} \mathbf{x} + \mathbf{b}_v^{m,\ell}$$

$$\mathbf{h}_2^\ell = \text{Dropout}(\mathbf{W}_{m_1}^\ell \cdot \mathbf{h}_1^\ell + \mathbf{b}_{m_1}^\ell) \quad (1)$$

$$\mathbf{h}_3^\ell = \mathbf{g}_{LN_1}^\ell \odot \frac{(\mathbf{h}_2^\ell + \mathbf{x}) - \mu}{\sigma} + \mathbf{b}_{LN_1}^\ell \quad (2)$$

$$\mathbf{h}_4^\ell = \text{GELU}(\mathbf{W}_{m_2}^\ell \cdot \mathbf{h}_3^\ell + \mathbf{b}_{m_2}^\ell) \quad (3)$$

$$\mathbf{h}_5^\ell = \text{Dropout}(\mathbf{W}_{m_3}^\ell \cdot \mathbf{h}_4^\ell + \mathbf{b}_{m_3}^\ell) \quad (4)$$

$$\text{out}^\ell = \mathbf{g}_{LN_2}^\ell \odot \frac{(\mathbf{h}_5^\ell + \mathbf{h}_3^\ell) - \mu}{\sigma} + \mathbf{b}_{LN_2}^\ell \quad (5)$$



# BitFit

BitFit tunes only the bias terms in self-attention and MLP layers.

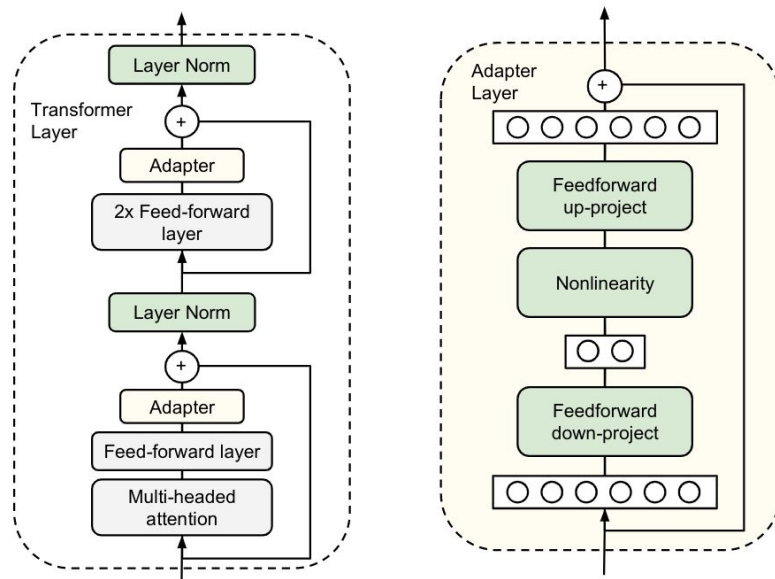
[Logan et al. \(2021\)](#) and [Mahabadi et al. \(2022\)](#) find that prompt-based finetuning with BitFit performs on-par with or better than full prompt-based finetuning on few-shot tasks.

# Adapters

Add MLP + skip connection after each feedforward layer.

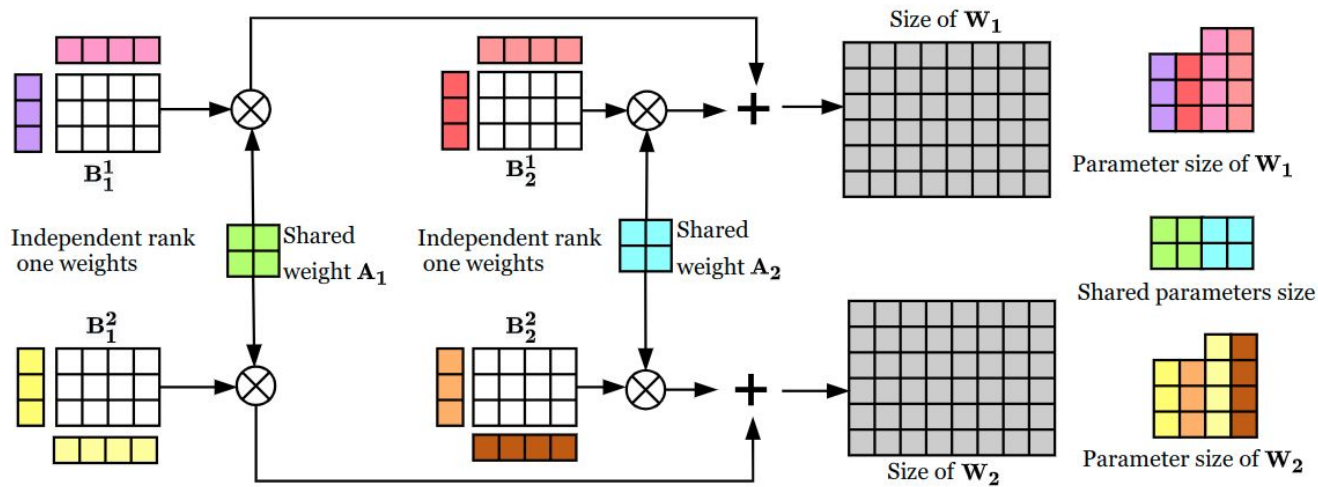
MLP projects to a low dimensional space to reduce parameters.

Tune only the MLP layers on new tasks.



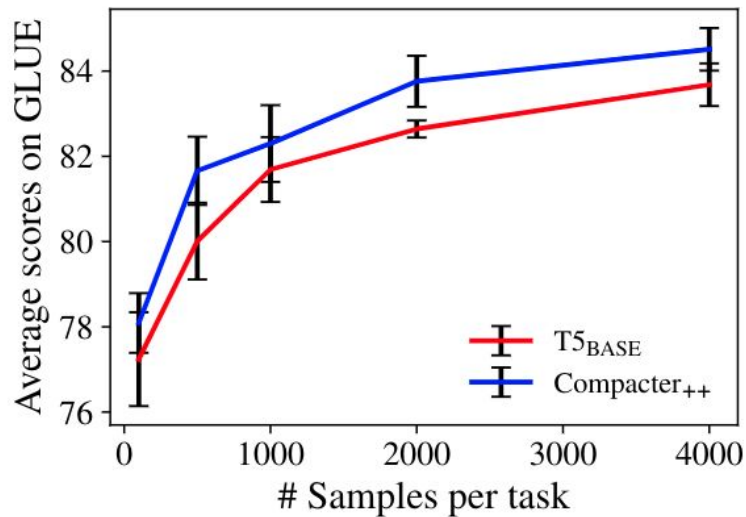
# Compacter

Compacter layers modify Adapter layers to use low-rank parameterized hypercomplex multiplication (i.e., parameter efficient  $W$ 's).



# Compacter

Compacter layers modify Adapter layers to use low-rank parameterized hypercomplex multiplication (i.e., parameter efficient W's).

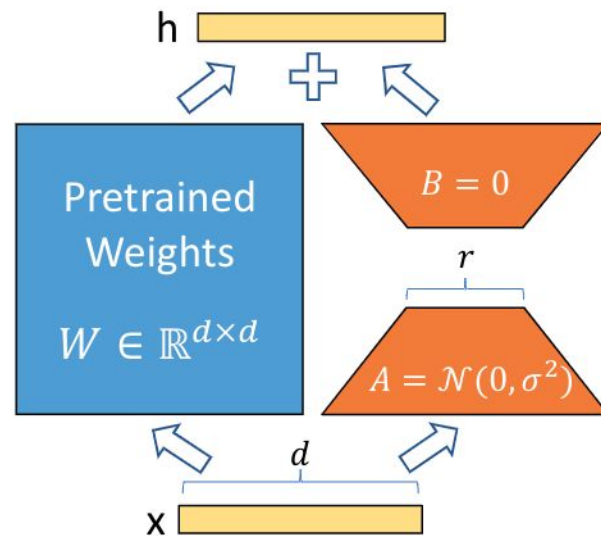


# LoRa

Low rank additive updates to model weights.

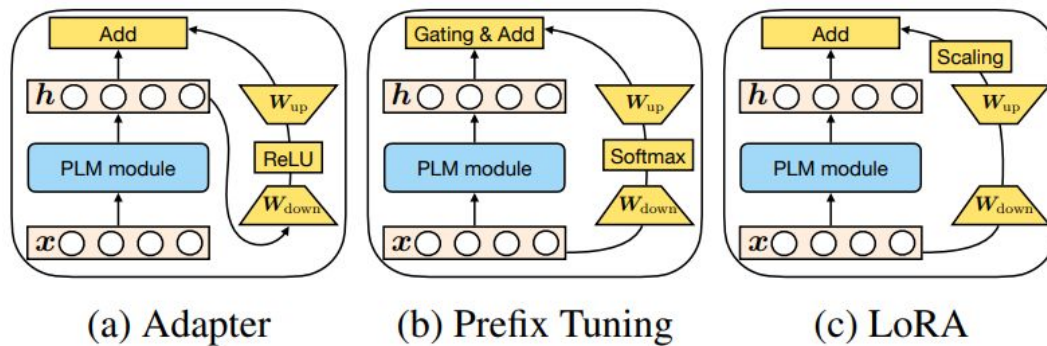
$$W = W_0 + AB$$

Where the rank of A and B  $\ll \min(d, h)$



# Towards a Unified View of Parameter-Efficient Transfer Learning

Approaches are more similar than at first glance.



# (IA)<sup>3</sup>

## Infused Adapter by Inhibiting and Amplifying Inner Activations

A new form of parameter-efficient fine-tuning. A good param efficient finetuning method should have the following properties:

- Add/update as few params as possible

- Strong performance after few-shot training on new tasks

- Allow use for mixed-task batches

  - Ideally shouldn't modify the model

  - Instead directly modify the activations

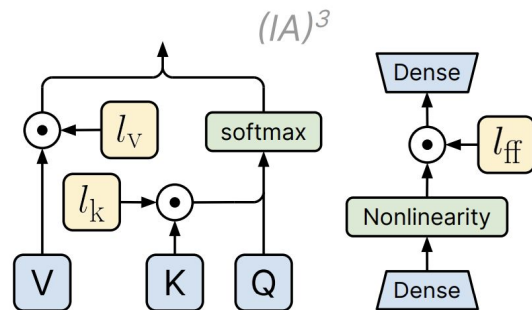
# $(IA)^3$

Element-wise rescaling of model activations with a learned vector:

keys and values in self-attention

Keys and values in encoder-decoder attention

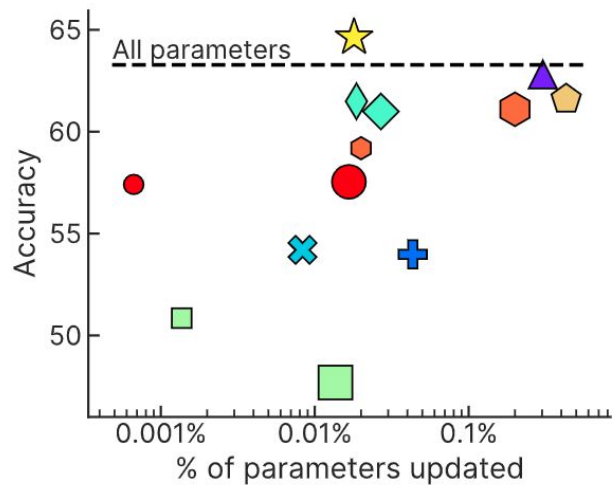
Intermediate activation of the position-wise  
feed-forward networks



Mixed-task batches are possible because each sequence of activations in the batch can be separately and cheaply multiplied by its associated learned task vector.

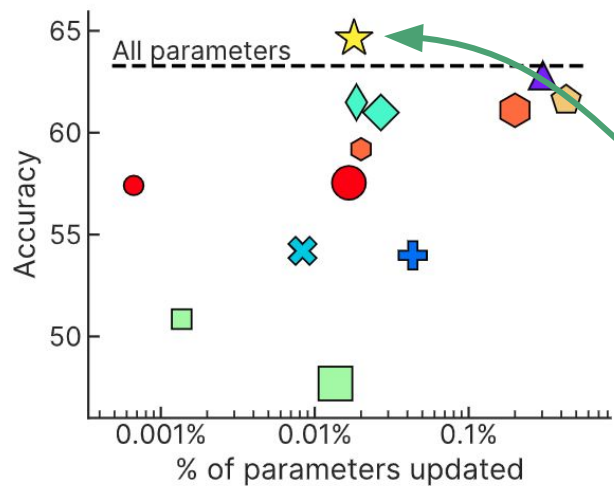


# Comparison of Parameter Efficient Finetuning Methods



[Liu et al., 2022](#). "Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning"

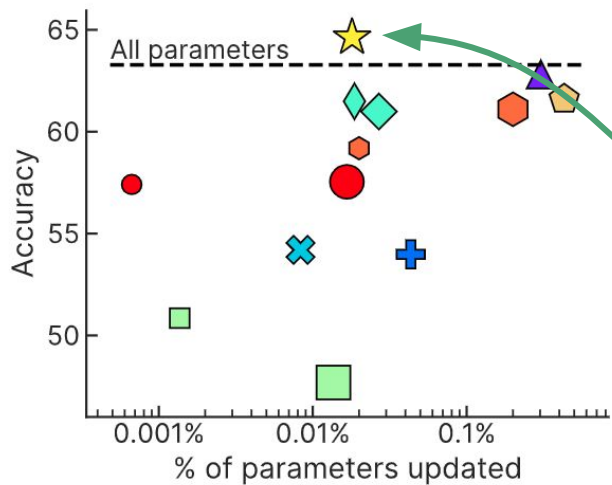
# Comparison of Parameter Efficient Finetuning Methods



(IA)<sup>3</sup> outperforms



# Comparison of Parameter Efficient Finetuning Methods



Can be further improved using meta-training.

Watch out for T-Few in the second half of the tutorial!



# Summary of gradient-based LM task adaptation

- Prompt-based finetuning much better than traditional approach
  - Can be further improved with unlabeled data using PET
  - Sensitive to choice of label tokens, somewhat to prompt
- More accurate than in-context learning with much smaller models
  - 'Nuff said
- Parameter-efficient alternatives can increase accuracy
  - Seems to rely on altering model's internals, even if only bias terms
  - Input-level methods relatively less effective

# Open questions & future work

- How should prompts affect prompt-based finetuning?
  - How can we make the effect of good prompts larger?
  - Alternatively, should unrelated/nonsensical prompts perform worse?
- Can discrete prompt search methods (e.g., AutoPrompt and LM-BFF) find good initializations prompt tuning?
- Most existing work focuses on classification, do results still hold on generation tasks?
- Can we improve efficiency even further?
  - DistilBERT-sized models for few-shot learning?

# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation benchmark [Arman]
- 17:05–17:25 Part 6: Meta-training [Arman]
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]

# Questions?

---

Slides available at:

<https://github.com/allenai/acl2022-zerofewshot-tutorial>

# Break

Come back at 16:30 Irish Time

---

Slides available at:

<https://github.com/allenai/acl2022-zerofewshot-tutorial>



# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation benchmark [Arman]
- 17:05–17:25 Part 6: Meta-training [Arman]
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]

# Part 4: Other methods for defining a task

---

Slides available at:

<https://github.com/allenai/acl2022-zerofewshot-tutorial>

# So far..

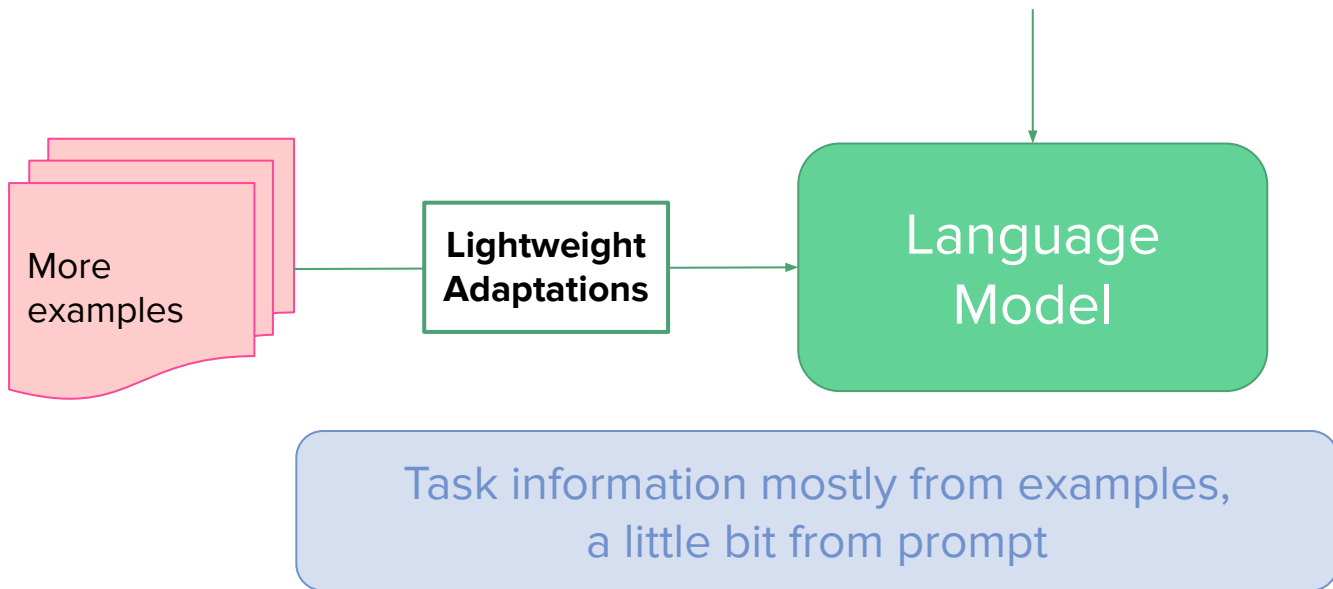
**In-Context Learning:**

An effortlessly accomplished and richly resonant work. **It was great!**

A mostly tired retread of several other mob tales. **It was terrible!**

**Prompting:**

A three-hour cinema master class. **It was** \_\_\_\_\_ (great/terrible)



# The Turking Test

Instead of example, give them “instructions”

Turking Tasks: annotate NewsQA

- given annotation guidelines

**Listing Nouns:** from a given sentence

**Retrieve an Element by Index**

- $n$ th word or character in a sentence

Write questions about the highlights of a story.

Steps

1. Read the highlights
2. Write questions about the highlights

Example

Highlights

- Sarah Palin from Alaska meets with McCain
- Fareed Zakaria says John McCain did not put country first with his choice
- Zakaria: This is “hell of a time” for Palin to start thinking about national, global issues

Here are

• Math g

• Pi, or r

of a circ

• The Pi

• An

• Al

• Al

• Al

• Al

• Al

• Al

• Al

• Al

• Al

• Al

• Al

• Al

• Al

Questions

• What

• What

• What

• What

• What

• What

• What

• What

• What

• What

• What

• What

• What

• What

• What

• What

Performed  
really bad!

• Do not re-use the same or very similar questions.

• Questions should be written to have short answers.

• Do not write “how” nor “why” type questions since their answers are not short. “How far/long/many/much” are okay.

# ZERoShot learning from TAsk descriptions

Instead of instructions, use QA

100 information seeking *tasks*

Train it on some, test on others



"What camp zones are in this national park?"

"Does this national park have stores that sell firewood?"

"Does this national park have a gift shop selling handmade items?"

"Where are bird watching spots near a lake in this national park?"

"What are the popular activities to do in the rivers at this national park?"

"Is spelunking at this national park allowed?"

"Can you boat and grill at this national park?"

"How many people can fit in group campsites in this national park?"

"How long is the cave in this national park?"

"Could you mention the camp zones in this national park?"

"How many plants living inside this national park are endangered?"

No examples at test-time  
(Zero-shot!)

# Natural Instructions (v1)

Give detailed human-readable instructions (that contain examples)

**Input:** *She chose to make a salad for lunch on Sunday.  
Question: how long did it take for her to make a salad?*

*tagging  
essential  
phrases*

**Crowdsourcing Instruction:** *List all the words that are essential for answering it correctly. [...]*

**Output:**  
*making  
salad*

*answering  
questions*

**Crowdsourcing Instruction:**  
*Answer the provided question based on a given [...]*

**Output:**  
*30mins*



# Natural Instructions (v1)

## Structured Schema for Instructions

Title, Definition, Emphasis/Caution,  
Things to avoid, +/- examples, prompt

- **Title:** Modifying a fill in the blank question on persons
- **Definition:** You're given a fill-in-the-blank question where the answer is PersonX. You need to minimally change the given question so that the answer flips to PersonY. This task typically involves replacing one word i.e. the 'trigger word' by its antonym (e.g. changing from "sympathetic" to "stern").
- **Emphasis & Caution:** 1. Your question must contain at least 15 and at most 30 words. 2. Your question must have atleast 70% overlapping words with the given question 3. Your question must contain only one blank. 4. Make sure that PersonX and PersonY have the same gender. 6. In your question, PersonX and PersonY should be used only ONCE and PersonX should appear earlier than PersonY. [...]
- **Things to avoid:** 1. You should not change any content in the given question beyond a word or two i.e. the trigger word/phrase. [...]

### Positive Example

- **Input:** Context word: upset. Question: PersonX yelled at PersonY because \_ was so upset about the news. Answer: PersonX.
- **Output:** PersonX comforted at PersonY because \_ was so upset about the news.
- **Reason:** On replacing the trigger word "yelled" by its antonym "comforted", the answer flips to PersonY which is as per the given instruction. So, this is a valid question.

### Negative Example

- **Input:** Context word: step. Question: PersonX was always ahead of PersonY, as \_ walked with a quick step. Answer: PersonX.
- **Output:** PersonY was always ahead of PersonY, as \_ walked with a quick step.
- **Reason:** Here, the issue is that the usage order of PersonX and PersonY has been changed in the generated question. Remember that, for a question to be valid, PersonX should appear earlier than PersonY.
- **Suggestion:** -

- **Prompt:** What is the type of the answer corresponding to the given question? Number, Date, or Span?

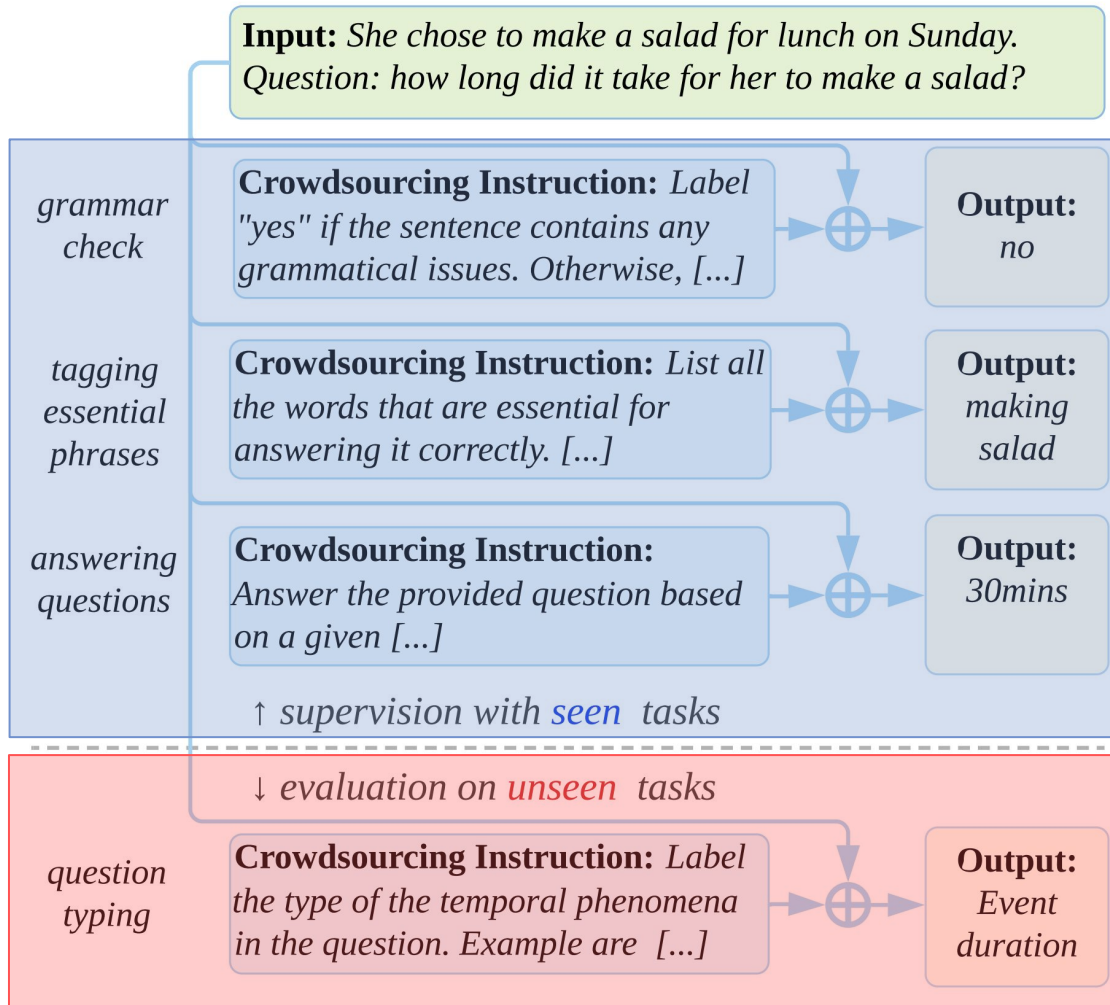
### task instance

- **Input:** Context Word: day. Question: PersonX learned new organizational skills from PersonY because \_ 's day schedule was very chaotic. Answer: PersonX
- **Expected Output:** PersonX learned new organizational skills from PersonY because \_ 's day schedule was very efficient.

# Need to train

Instructions gathered for 61 tasks

- split into **seen** and **unseen**
- Not zero-shot





# Concurrent work: T0 and FLAN

- Apply this idea to many more tasks in NLP
- Introduced notion of task categories, a *task* is a group of datasets
- **Unseen** datasets should belong to the same *task*, i.e. never see any NLI
- Multiple prompts for each task/dataset

## FLAN; [Wei et al ICLR 2022](#)

- Hold out one group at a time
- 10 prompts per task
- Very large model (137B)

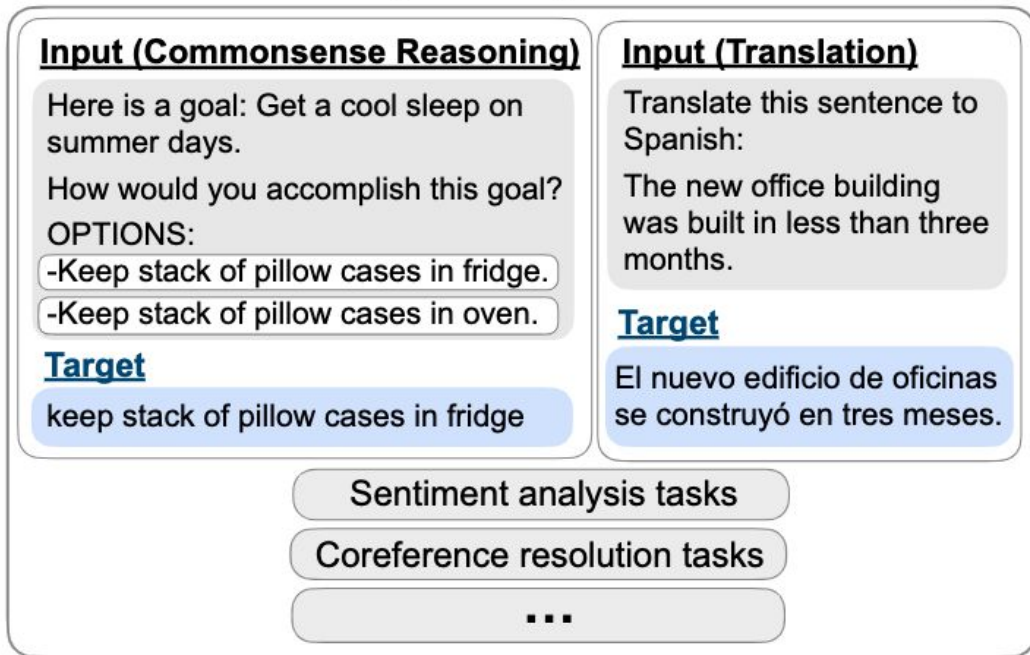
## T0; [Sanh et al ICLR 2022](#)

- Hold out 5 groups (23 datasets)
- More diversity, 12 prompts per task
- Much smaller model (11B)

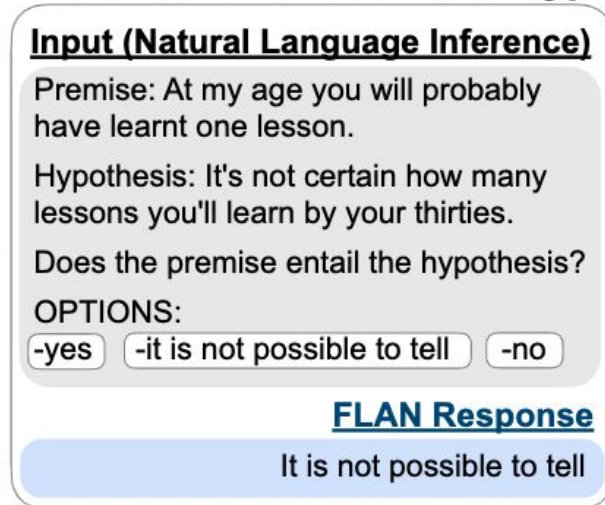
We'll hear a lot more about them by Arman in Section 6

# Example of Instructions (from FLAN)

## Finetune on many tasks (“instruction-tuning”)



## Inference on unseen task type



# PromptSource (and P3)

P3: Public Pool of Prompts, now 2085 prompts on 183 datasets

Dataset ?

No of prompts created for **cosmos\_qa** : 13

Prompt name ?

cosmos\_qa |

description\_context\_question\_text | ▾

- context\_answer\_to\_question
- context\_description\_question\_ans...
- context\_description\_question\_ans...
- context\_description\_question\_text
- context\_question\_description\_ans...
- context\_question\_description\_ans...
- context\_question\_description\_text
- description\_context\_question\_ans...

cord19

cornell\_movie\_dialog

cos\_e

cosmos\_qa

covid\_qa\_castorini

covid\_qa\_deepset

covid\_qa\_ucsd

Input template

```
Read the following context and answer the question.
Context: {{ context }}
Question: {{ question }}
Answer:
```

Target template

```
{{ answer_choices[label] }}
```

<https://github.com/bigscience-workshop/promptsources>  
<https://huggingface.co/datasets/bigscience/P3>

# Natural Instructions (v2)

- 1616 tasks across 76 task types
- Simplified schema
- Many languages: 576 non-English
- More domains and reasoning types
- <https://instructions.apps.allenai.org/>

More tasks help  
Larger models help  
More instances don't really help

## Task Instruction

### Definition

"... Given an utterance and recent dialogue context containing past 3 utterances (wherever available), output 'Yes' if the utterance contains the small-talk strategy, otherwise output 'No'. Small-talk is a cooperative negotiation strategy. It is used for discussing topics apart from the negotiation, to build a rapport with the opponent."

### Positive Examples

- **Input:** "Context: ... 'That's fantastic, I'm glad we came to something we both agree with.' Utterance: 'Me too. I hope you have a wonderful camping trip.'"
- **Output:** "Yes"
- **Explanation:** "The participant engages in small talk when wishing their opponent to have a wonderful trip."

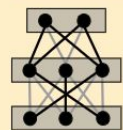
### Negative Examples

- **Input:** "Context: ... 'Sounds good, I need food the most, what is your most needed item?!' Utterance: 'My item is food too'."
- **Output:** "Yes"
- **Explanation:** "The utterance only takes the negotiation forward and there is no side talk. Hence, the correct answer is 'No'."

## Evaluation Instances

- **Input:** "Context: ... 'I am excited to spend time with everyone from camp!' Utterance: 'That's awesome! I really love being out here with my son. Do you think you could spare some food?'"
- **Expected Output:** "Yes"

### Tk-Instruct



# Summary of Training with Instructions

Name	Variety of Task Types	Number of Tasks	Instructions per Task	Held out Tasks
<b>Turking</b>	3 types + synthetic	3 tasks	1 per task	All of them
<b>ZEST</b>	Machine comprehension	100 tasks 3 domains	1 per task, but with perturbations	40 tasks at random
<b>Natural Instructions</b>	6 types (all related to MR)	61 tasks	1 per task	Multiple setups: Max 12 tasks
<b>FLAN *</b>	12 types	62 tasks	10 per task	5-10 tasks / 1 type
<b>T0 *</b>	13 types	62 tasks (now 176)	11-12 per task (now 2052)	23 tasks / 5 types
<b>Natural Instructions (v2)</b>	76 types	1616 tasks	1 per task	154 tasks / 12 types

\* More details later!

# Some related directions

## InstructGPT: [Ouyang et al 2021](#)

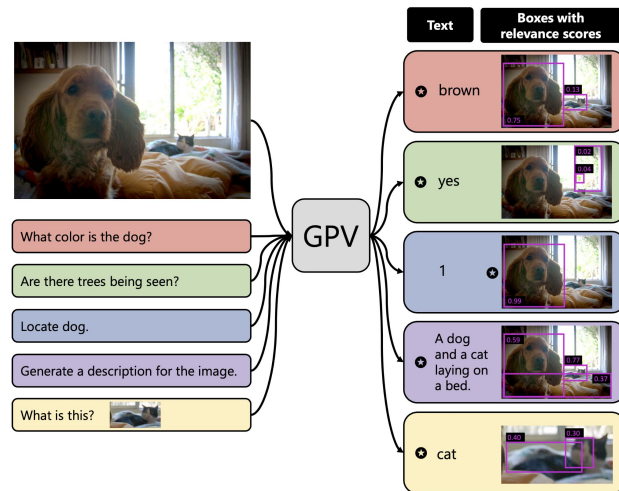
- Train with instructions, more annotations
- additional losses on the output using RL

## Learning from Explanations

- Free text explanations: [Camburu et al NeurIPS 18](#)
- Formalization: [Hase & Bansal ACL LNLS 2022](#)
- Chain-of-thought prompting [next]

## Other modalities

- Robots with instructions e.g. [Zhao et al EACL 2021](#)
- Vision tasks as VQA e.g. [Gupta et al CVPR 2022](#)



# Chain of Thought Prompting

Give more “instructions” specific to the instance, only in-context learning

## Standard prompting

**Input:** Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

...

Q: John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs?

A:

**Model output:** The answer is 50. ❌

## Chain of thought prompting

**Input:** Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

...

Q: John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs?

A:

**Model output:** John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. So that is  $10 \times .5 = 5$  hours a day. 5 hours a day  $\times 7$  days a week = 35 hours a week. The answer is 35 hours a week. ✅

# Summary of Learning from Task Descriptions

- Alternate **form of supervision**
  - Not just input/output examples
  - Not just a “format” of the task
- Often useful with **finetuning and multi-tasking**
  - For new, unseen task, they are able to do better
- Can encompass both **Zero and Few shot learning**
  - Are you putting examples in the instructions or not?
- **Scalability of gathering instructions** is a concern
  - P3 and Natural Instructions (v2) are useful in addressing this
- Currently, **promising practical approach** to few-shot learning!
  - What other ways can we describe tasks?



# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation benchmark [Arman]
- 17:05–17:25 Part 6: Meta-training [Arman]
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]

# Part 5: Evaluation Benchmarks

---

Slides available at:

<https://github.com/allenai/acl2022-zerofewshot-tutorial>

# Introduction

Evaluation of zero- and few-shot learning capabilities of models can be challenging

In this section we overview evaluation approaches and benchmarks commonly used in the literature

We provide advantages and disadvantages of each the approaches and discuss best practices of designing new benchmarks

# Evaluation in few-shot setting

- How to reliably evaluate models on zero-/few-shot setting?
- **Sampling** from larger existing datasets
- e.g., SuperGLUE → FewGLUE (Schick and Schütze, 2020)
  - 32 examples from SuperGLUE

# Evaluation in few-shot setting

High variance of results when changing the training set

<b>Model</b>	<b>CB</b> Acc. / F1	<b>RTE</b> Acc.	<b>MultiRC</b> EM / F1a	<b>Avg</b> –
GPT-3	82.1 / 57.2	<b>72.9</b>	32.5 / 74.8	65.4
PET $\neg$ dist ( $\Sigma_0$ )	83.9 / 76.2	66.4	38.9 / 76.2	<b>68.0</b>
PET $\neg$ dist ( $\Sigma_1$ )	82.1 / 57.4	61.4	<b>39.2 / 77.9</b>	63.2
PET $\neg$ dist ( $\Sigma_2$ )	<b>87.5 / 84.0</b>	61.4	34.7 / 76.3	67.6

Table 6: Results on selected tasks for GPT-3 and for PET using training sets  $\Sigma_0, \Sigma_1, \Sigma_2$

Schick and Schütze (2020), It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners

# Evaluation in few-shot setting

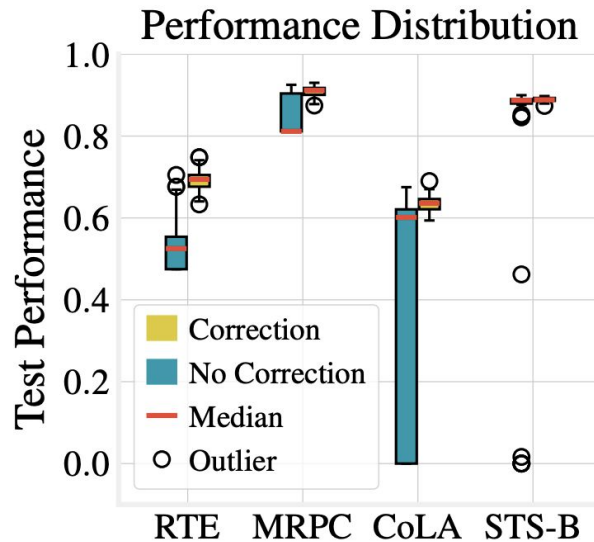
Early works suggest using Small **fixed** sets (e.g., 32 fixed examples)

Assumption: Since all models use the same small set, the comparisons are valid

# Small fixed sets

Problem: Randomness.

Instability based on different seeds

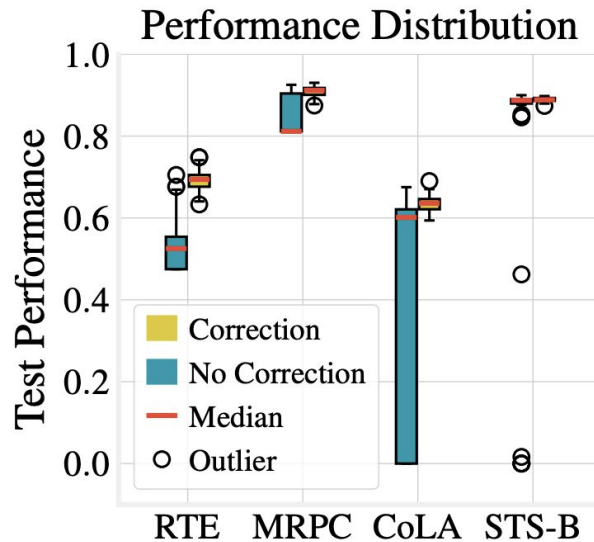


[Revisiting Few-sample BERT Fine-tuning](#), Zhang et al (2021)

# Small fixed sets

Problem: Randomness.

Instability based on different seeds



[Revisiting Few-sample BERT Fine-tuning](#), Zhang et al (2021)



# Sampling multiple sets

Instead of one fixed set, sampling multiple small sets. [LM-BFF](#) (Gao et al., 2021)

Measure average/stdev of performance

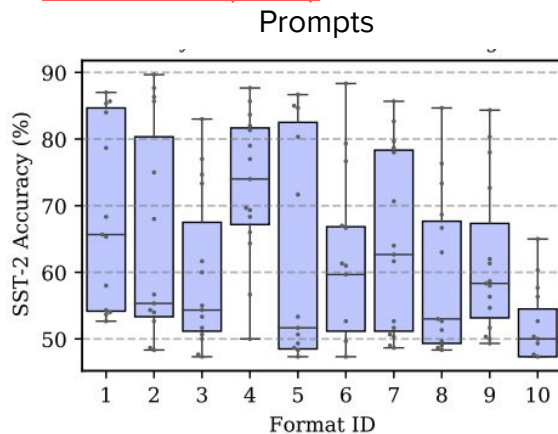
Advantage: we can see variance

Many papers follow a similar setup.

# Evaluation

Few-shot performance depends heavily on various factors

[Perez et al., \(2021\)](#), *True Few-Shot Learning with Language Models*



[Zhao et al., \(2021\)](#) Calibrate Before Use:  
Improving Few-Shot Performance of Language  
Models

[Lu et al., \(2022\)](#) Fantastically Ordered Prompts  
and Where to Find Them: Overcoming Few-Shot  
Prompt Order Sensitivity

[Tam et al., \(2021\)](#) Improving and Simplifying  
Pattern Exploiting Training

# Evaluation scenarios

Learning Scenario	Many Train Distributions	Many Train Examples	Many Val Examples
Data-Rich Supervised	x	✓	✓
Multi-Dist, Few-Shot	✓	x	x
Tuned Few-Shot	x	x	✓
True Few-Shot	x	x	x

*[Perez et al., \(2021\)](#), True Few-Shot Learning with Language Models*

# Evaluation scenarios

Conventional  
Methods



Learning Scenario	Many Train Distributions	Many Train Examples	Many Val Examples
Data-Rich Supervised	x	✓	✓
Multi-Dist, Few-Shot	✓	x	x
Tuned Few-Shot	x	x	✓
True Few-Shot	x	x	x

*[Perez et al., \(2021\)](#), True Few-Shot Learning with Language Models*

# Evaluation scenarios

Prototypical Nets,  
Matching Nets, ←  
Meta-learning  
methods, etc

Learning Scenario	Many Train Distributions	Many Train Examples	Many Val Examples
Data-Rich Supervised	x	✓	✓
Multi-Dist, Few-Shot	✓	x	x
Tuned Few-Shot	x	x	✓
True Few-Shot	x	x	x

*[Perez et al., \(2021\)](#), True Few-Shot Learning with Language Models*

# Evaluation scenarios

GPT-3, Adapet,  
CLIP, ...



Learning Scenario	Many Train Distributions	Many Train Examples	Many Val Examples
Data-Rich Supervised	x	✓	✓
Multi-Dist, Few-Shot	✓	x	x
Tuned Few-Shot	x	x	✓
True Few-Shot	x	x	x

*[Perez et al., \(2021\)](#), True Few-Shot Learning with Language Models*

# Evaluation scenarios

Learning Scenario	Many Train Distributions	Many Train Examples	Many Val Examples
Data-Rich Supervised	x	✓	✓
Multi-Dist, Few-Shot	✓	x	x
GPT-3, Adapet, CLIP, ... ←	x	x	✓
True Few-Shot	x	x	x

Many works are evaluated on this setting

When additional validation examples are provided -> The value of few-shot learning is not there

*[Perez et al., \(2021\)](#), True Few-Shot Learning with Language Models*

# Evaluation scenarios

Learning Scenario	Many Train Distributions	Many Train Examples	Many Val Examples
Data-Rich Supervised	x	✓	✓
Multi-Dist, Few-Shot	✓	x	x
Tuned Few-Shot	x	x	✓
True Few-Shot	x	x	x

No additional  
large validation  
examples





# Evaluation scenarios

In true few-shot learning model selection should be done using the available few-shot data.

Example model selection methods from Perez et al., (2021):

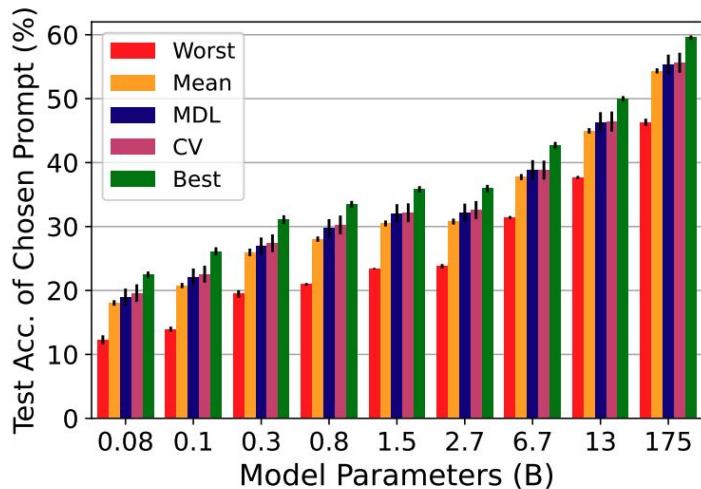
Cross validation

Validation on fold  $k$ , other folds used for training

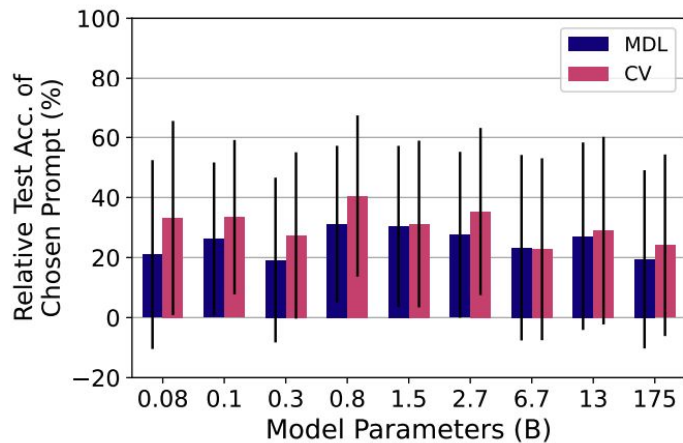
Minimum Description Length (MDL)

Validation on fold  $k$ , previous  $k-1$  folds used for training

# Evaluation under true-few shot



Accuracy of CV/MDL-chosen prompts vs. accuracy of the worst, average (randomly-selected), and best prompt (prior work)

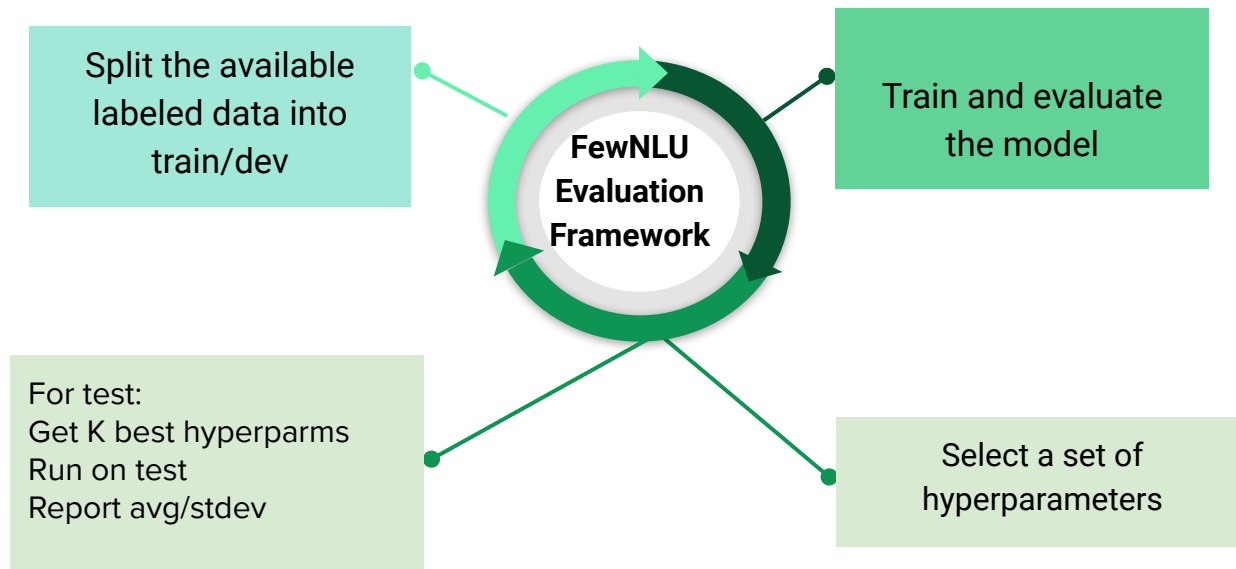


The average accuracy gain from using CV/MDL-chosen prompts instead of randomly-chosen ones, relative to the gain from the best prompt

# FewNLU

Similarly advocates for model selection using the small labeled set

Evaluation framework consisting of the following repeated episodic procedure



# FewNLU

Strategies to split the few-shot training data into train/validation sets

Considerations for constructing the data splits:

**1- Performance:** how to split the data to achieve good test set performance

**2- Correlation between dev/test:** Ideally there should be good correlation between the small dev set performance and the test set over distribution of hyperparameters

**3- Stability:** The number of runs should have small impact on the above metrics

# FewNLU - data splitting strategies

**1- K-fold Cross Validation** (Perez et al., 2021)

**2- MDL** (Minimum description length) (similar to Perez et al., 2021)

Evaluate on fold K, use \*previous\* K-1 folds as training

**3- Bagging**

Sample x% of the labeled data with replacement as training and others as validation

**4- Random Sampling**

**5- Model informed splitting** (Cluster to two sets using BERT-base CLS representation)

**6- Multi-Splits**

Split data into two parts using a fixed split ratio (they use half)

# FewNLU - data splitting strategies

In general the simple multi-split approach works well

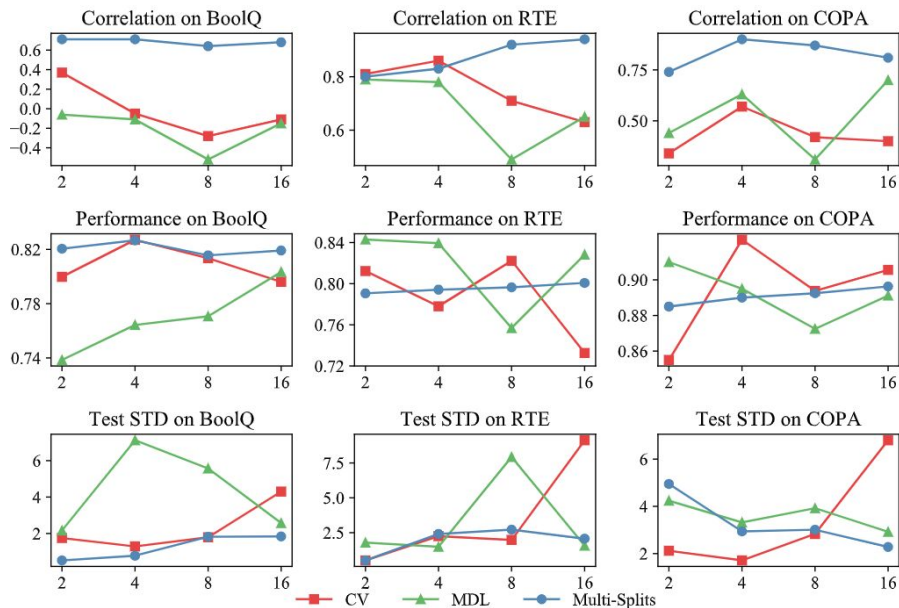
	Avg.
CV	78.72
MDL	77.00
BAG	77.62
RAND	76.89
MI	75.44
MS	79.00

Average performance

	Avg.
CV	0.4427
MDL	0.3685
BAG	0.6716
RAND	0.6131
MI	0.5662
MS	0.7190

Correlation results

# FewNLU - data splitting strategies



Test performance, correlation and standard deviation along with different K (number of runs). A straight line shows more stability.

# Cross-task generalization

How can we evaluate the ability of models to learn from other tasks in NLP?

**CrossFit** introduces a benchmark for this goal

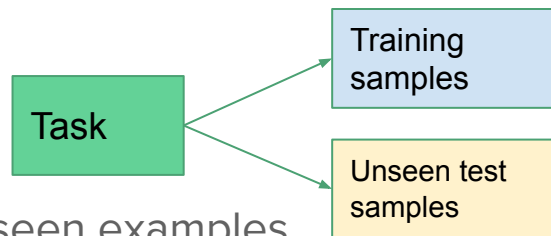
Compared with prior work, the new benchmark has diverse set of tasks



# Cross task generalization

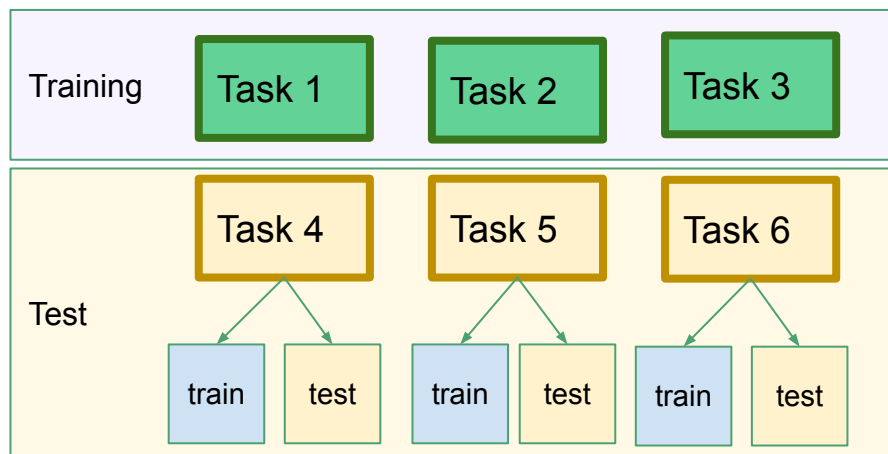
## Instance-level generalization

Generalize from few training examples to new unseen examples



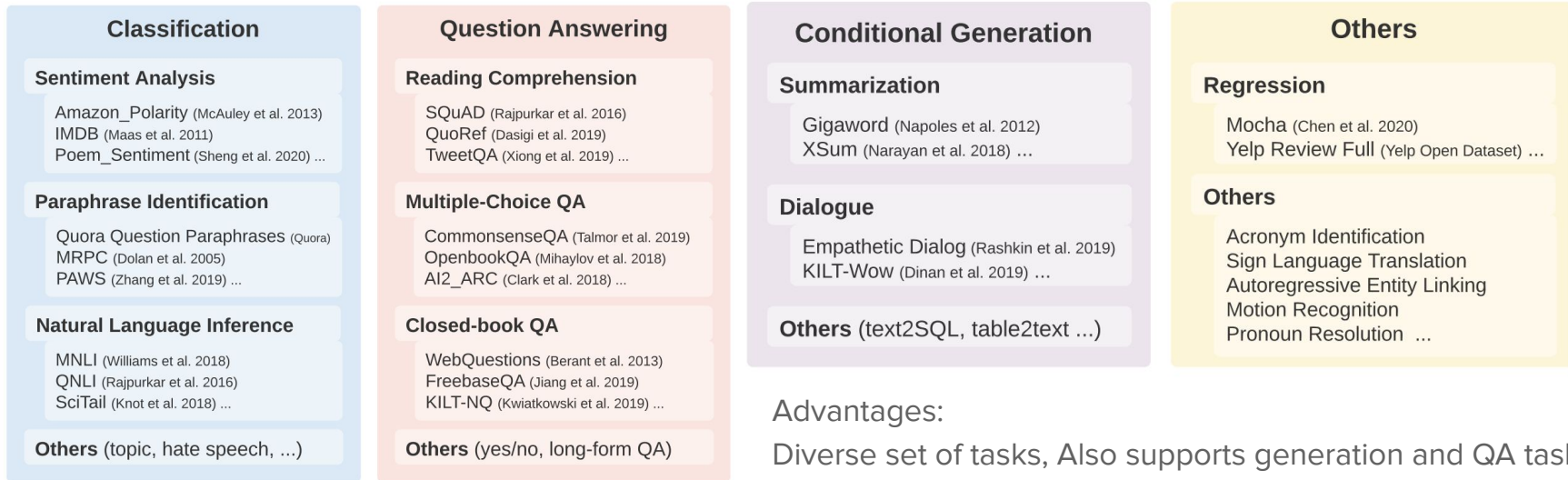
## Cross task generalization

Generalize from examples from seen tasks to new tasks with optional few training examples



# Cross task generalization

**CrossFit:** a set of **160** diverse tasks (Ye et al., 2021)



[Ye et al., \(2021\)](#) CROSSFIT : A Few-shot Learning Challenge for Cross-task Generalization in NLP

# Cross task generalization

## Evaluation metric

Average Relative Gain (**ARG**) to measure overall performance gain before and after training on other tasks. Example:

	Few-shot fine-tuning	Cross task training + few-shot fine-tuning	Gain
Task A	50.0	60.0	<b>+20%</b>
Task B	40.0	35.0	<b>-12.5%</b>

$$\mathbf{ARG = (+20 - 12.5) / 2}$$

# How should we design new few-shot benchmarks?

In a fast moving field it is difficult to have a benchmark that is useful for a long time

- Community overfits to the benchmark over time

- New tasks/datasets become available

New benchmarks will be released all the time

Are there any principles that we can follow to ensure building reliable benchmarks?

# How should we design new few-shot benchmarks?

[Bragg et al., \(2021\)](#) introduce **FLEX Principles**

A set of requirements and best practices for designing rigorous and reliable few-shot benchmarks and evaluation frameworks

# FLEX Principles ([Bragg et al., 2021](#))

A few-shot evaluation benchmark should:

## 1 - Represent diversity of **generalization types**:

- ❑ **Pre-training generalization**
- ❑ **Cross domain generalization**
- ❑ **Cross class generalization**
- ❑ **Cross task generalization**

This supports studying and analyzing different types of few-shot generalization

# Unified evaluation for all types of generalizations

1- Pre-training generalization



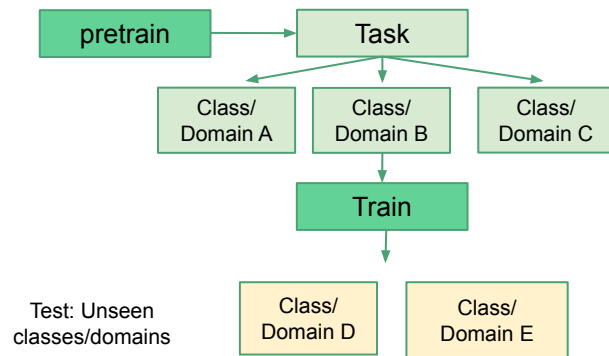
# Unified evaluation for all types of generalizations



1- Pre-training generalization

2- Cross class generalization

3- Cross domain generalization



[Bragg et al \(2021\)](#) FLEX: Unifying Evaluation for Few-Shot NLP



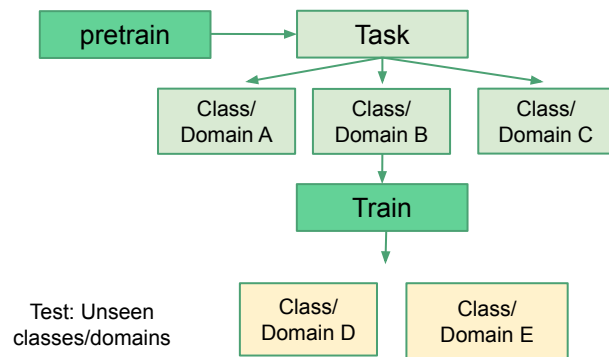
# Unified evaluation for all types of generalizations

1- Pre-training generalization

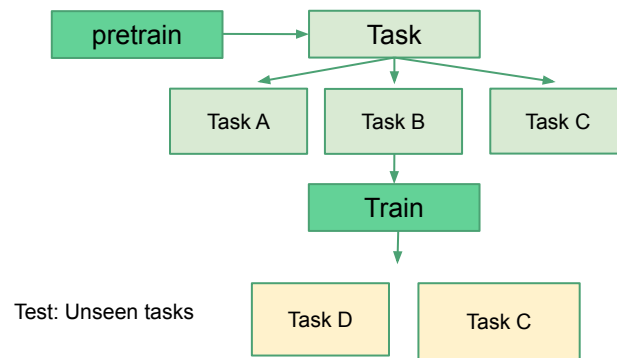


2- Cross class generalization

3- Cross domain generalization



4- Cross task generalization



[Bragg et al \(2021\)](#) FLEX: Unifying Evaluation for Few-Shot NLP

# FLEX Principles ([Bragg et al., 2021](#))

A few-shot evaluation benchmark should:

**1** - Represent diversity of **generalization types**.

**2**- Contain real-world dataset challenges

- ❑ **Variable** number of classes & number of examples per class
- ❑ **Unbalanced** training sets
- ❑ **Zero-shot** (support zero-shot evaluation)
- ❑ **No extra validation data** for hyperparameter or prompt tuning

# FLEX Principles ([Bragg et al., 2021](#))

A few-shot evaluation benchmark should:

- 1** - Represent diversity of **generalization types**.
- 2**- Contain real-world dataset challenges
- 3**- Design and reporting should follow careful statistical considerations

# Statistical considerations

- NLP few-shot evaluation often ignores statistical validity
- Arbitrary number of test episodes (number of test sets sampled for reporting)

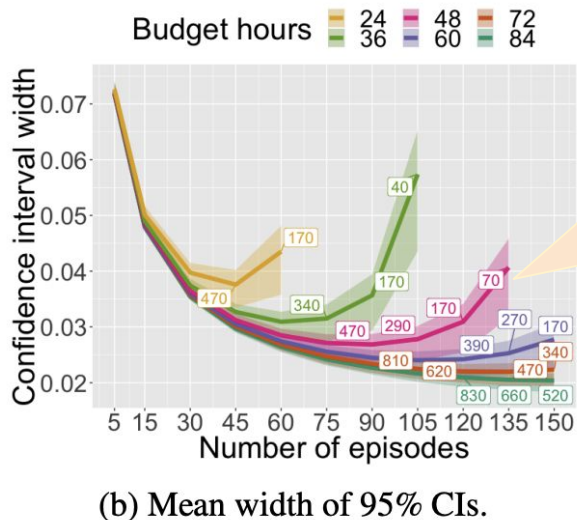
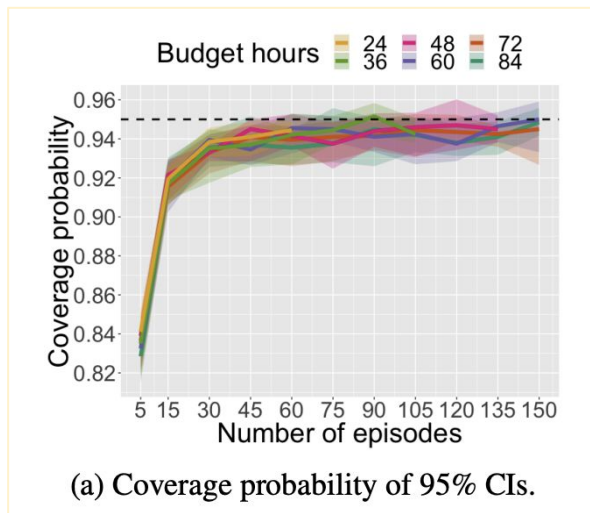
	CrossFit[71]	LM-BFF[23]	GPT-3[9]	DS[4]	SMLMT[3]	FewGlue[51]
# test episodes	5	5	1	1000	10	1

- What are the main considerations:
  - More episodes → More estimates on which we can compute `mean()` and `std()`
  - More episodes → More costly compute, especially for fine-tuning methods

**Unclear how reliable are the comparisons under each #test episode choice**

# Statistical considerations

A proper benchmark should produce performance estimates that are accurate, close to the true value, precise, low variance.



Each point is a chosen  
(*num\_episodes*, *test\_set\_size*)

e.g. for **48 GPU-hours** budget,  
CI width is minimized at

- *num\_episodes* = 90
- *test\_set\_size* = 470

# FLEX Benchmark ([Bragg et al., 2021](#))

NLP Few-shot benchmark that follows all these principles.

	CrossFit[75]	LM-BFF[24]	GPT-3[10]	DS[5]	SMLMT[4]	FewGlue[56]	<b>FLEX (ours)</b>
Class Transfer	-	-	-	✓	-	-	✓
Domain Transfer	-	-	-	-	✓	-	✓
Task Transfer	✓	-	-	-	✓	-	✓
Pretraining Transfer	-	✓	✓	-	-	✓	✓
Shots per class	{16, 32}	16	variable	{1,5}	{4,8,16,32}	{total 32} <sup>5</sup>	[1–5]
Variable shots	-	-	✓	-	-	-	✓
Unbalanced	-	-	-	-	-	-	✓
Textual labels	✓	✓	✓	-	-	✓	✓
Zero-shot	-	✓	✓	-	-	-	✓
No extra test data	-	-	-	✓	✓	mixed <sup>6</sup>	✓
# test episodes	5	5	1	1000	10	1	90
Reporting	avg	avg, SD	avg	avg, SD	avg, SD	avg, SD	all <sup>7</sup>
# datasets	160	16	37	7	18	8	20

# RAFT - Real-world Few-shot classification benchmark

Most existing benchmarks focus on synthetic tasks

Key considerations

**Naturally occurring data:** move away from synthetic tasks

**Intrinsic value:** tasks that have intrinsic value like hate-speech detection, medical case report parsing and automated lit review

**Realistic class distributions:** Include imbalanced datasets

# RAFT - Real-world Few-shot classification benchmark

12 different tasks. examples:

- *ADE Corpus V2 (ADE)*. whether a medical sentence contains adverse drug effect
- *Banking77 (B77)*: online banking customer service queries with their intents
- *NeurIPS impact statement risks (NIS)*. annotate broader impact statements from NeurIPS papers based on possible harmful implications
- *Terms of Service (ToS)*: clauses from Terms of Services annotated whether they are unfair to customers
- *TweetEval Hate (TEH)*: hate-speech detection



# RAFT - Human performance

RAFT: at the time of publication

<b>Baseline</b>	<b>Avg</b>	<b>ADE</b>	<b>B77</b>	<b>NIS</b>	<b>OSE</b>	<b>Over</b>	<b>SOT</b>	<b>SRI</b>	<b>TAI</b>	<b>ToS</b>	<b>TEH</b>	<b>TC</b>
Human (crowdsourced)	<b>.735</b>	<b>.830</b>	<b>.607</b>	<b>.857</b>	<b>.646</b>	.917	<b>.908</b>	.468	.609	<b>.627</b>	<b>.722</b>	<b>.897</b>
GPT-3 (175B)	.627	.686	.299	.679	.431	<b>.937</b>	.769	<b>.516</b>	<b>.656</b>	.574	.526	.821
AdaBoost	.514	.543	.023	.626	.475	.838	.455	.506	.556	.560	.443	.625
GPT-Neo (2.7B)	.481	.452	.149	.408	.343	.681	.406	.493	.605	.565	.554	.636
GPT-2 (1.6B)	.458	.600	.121	.561	.245	.498	.380	.492	.612	.498	.311	.723
BART MNLI Zero-shot	.382	.234	.332	.615	.360	.462	.644	.026	.469	.122	.543	.400
Plurality class	.331	.446	.000	.353	.164	.337	.271	.493	.344	.471	.366	.391
GPT-3 Zero-shot	.292	.163	.000	.572	.323	.378	.628	.027	.362	.164	.303	.290

[Alex et al., \(2022\)](#) RAFT: A Real-World Few-Shot Text Classification Benchmark

# RAFT

The recent [T-Few](#) (Liu et al., May 2022) achieve human-level performance!  
(Details of T-Few in next part)

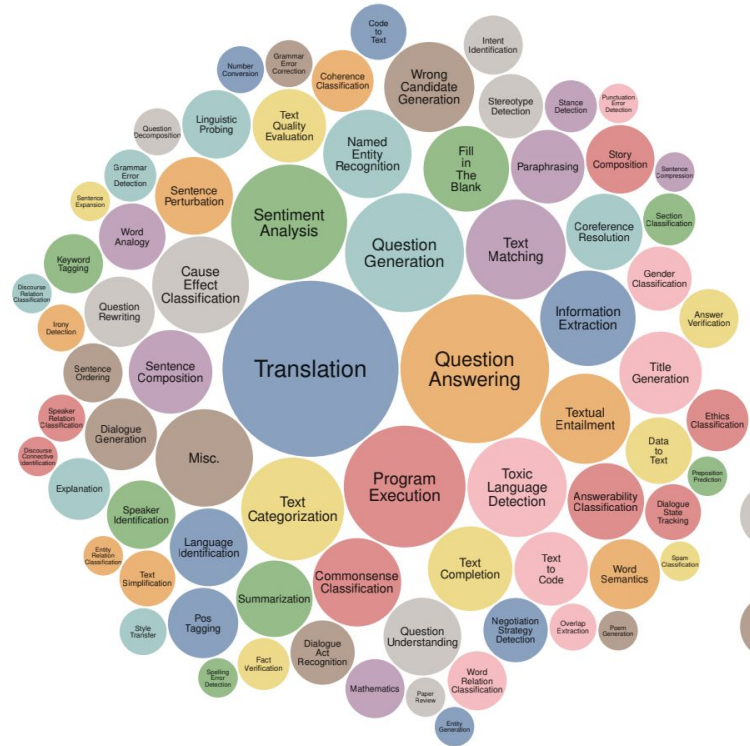
Method	Acc.
T-Few	75.8%
Human baseline <a href="#">[2]</a>	73.5%
PET <a href="#">[50]</a>	69.6%
SetFit <a href="#">[51]</a>	66.9%
GPT-3 <a href="#">[4]</a>	62.7%

# Other evaluation frameworks

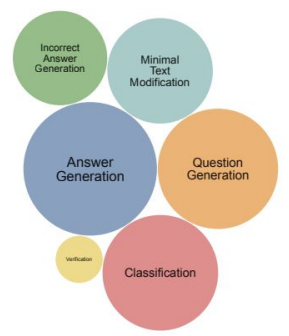
Recent papers used custom splits of existing datasets as their evaluations

FLAN, T0-Eval, BigBench, Natural-Instructions: Introduce large suite of tasks used for multitask prompted training

# Number of tasks



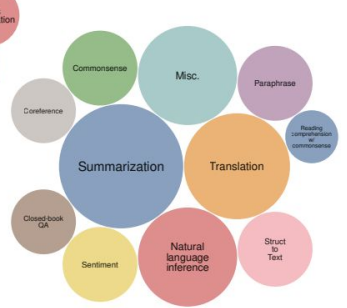
(a) NATINST<sub>v2</sub> (this work)



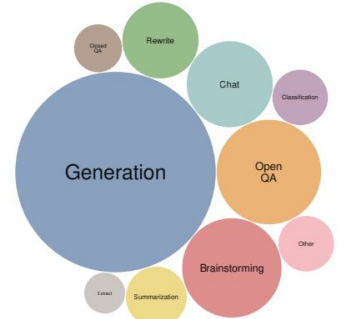
(b) NATINST



(c) PROMPTSOURCE (T0 subset)



(d) FLAN



(e) GPT3-INSTRUCT

[Wang et al \(2022\)](#) Benchmarking Generalization via In-Context Instructions on 1,600+ Language Tasks

# Other evaluation frameworks

## [EleutherAI Language Model Evaluation Harness](#) (EAI-Eval)

Focuses on prompt engineering

Hardcode a set of carefully designed prompts so that ICL is easy to evaluate

EAI-Eval only includes one prompt per task, whereas performance on T0-Eval is averaged over many prompts

# Summary and open problems

Thorough evaluation of few-shot learning methods is challenging

There's been many progress on creating better evaluation benchmarks for few-shot learning.

However, lack of unified, reliable, and robust benchmarks that is commonly used for evaluation sometimes makes it hard to measure true progress.

At the same time, it is challenging to address all evaluation needs in one unified benchmarks.

# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation benchmark [Arman]
- 17:05–17:25 **Part 6: Meta-training [Arman]**
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]

# Part 6: Meta-training

---

Continued training for improved zero/few-shot performance

Slides available at:

<https://github.com/allenai/acl2022-zerofewshot-tutorial>



# Meta-training - Introduction

This section focuses on approaches for continued training of language models for improving zero- and few-shot performance.

**Meta-training:** An additional training stage after the initial pretraining, utilizing supervised or self-supervised data.

In the literature, this is also referred to finetuning, continued pretraining, or intermediate pretraining/finetuning.

# Meta-training

LMs can perform few-shot learning. However:

There is discrepancy between the language modeling objectives used in pretraining and downstream task formats

Recall prompt engineering.

# Unifying pretraining and downstream formats

One way to address this issue is to unify pretraining and downstream task formats

Use a format that can naturally represent many NLP tasks (Q/A)

Meta-train the model on this new task format

[Unifew] [Bragg et al., \(2021\)](#) FLEX: Unifying Evaluation for Few-Shot NLP

[Meta-Tune] [Zhong et al., \(2021\)](#) Adapting Language Models for Zero-shot Learning by Meta-tuning on Dataset and Prompt Collections

# Unifying pre-training and downstream formats

## UniFew (Bragg et al., 2021)

- Convert all classification tasks to multiple choice QA
- Initialize the model with UnifiedQA (Khashabi et al., 2020) and continue meta-training
- This eliminates the need for complex tricks of prior work for finding the right prompt
- Model generates valid answer (no need to deal with constrained decoding methods)

```
Topic? \n (A) World (B) Sports (C) Business (D) Science \n
Wall St. Bears Claw Back Into the Black (Reuters) Reuters -
Short-sellers, Wall Street's dwindling\band of ultra-cynics, are
seeing green again.
Output: Business
```

# UniFew

One general prompt for each **input type** (4 total input types for task in FLEX).  
Most prior works engineer one unique and specific prompt for each **dataset**

Single text classification:

```
Topic?\n (A) Class1 (B) Class2 (C) Class3  
The document \n
```

Sentence-pair classification:

```
"Sentence 1" Is "Sentence 2"?\n  
(A) Yes (B) No (C) Maybe
```

Relation classification:

```
Mention-1 to mention-2? \n (A) Class1 (B) Class2 (C) Class3  
Some text #mention-1# some text *mention-2* some text
```

Entity recognition:

```
What is the type of the entity between the # marks? \n  
(A) Class1 (B) Class2 (C) Class3 \n  
Some text #mention-1# some text.
```

# UniFew

Initialize the model with UnifiedQA (Khashabi et al., 2020)

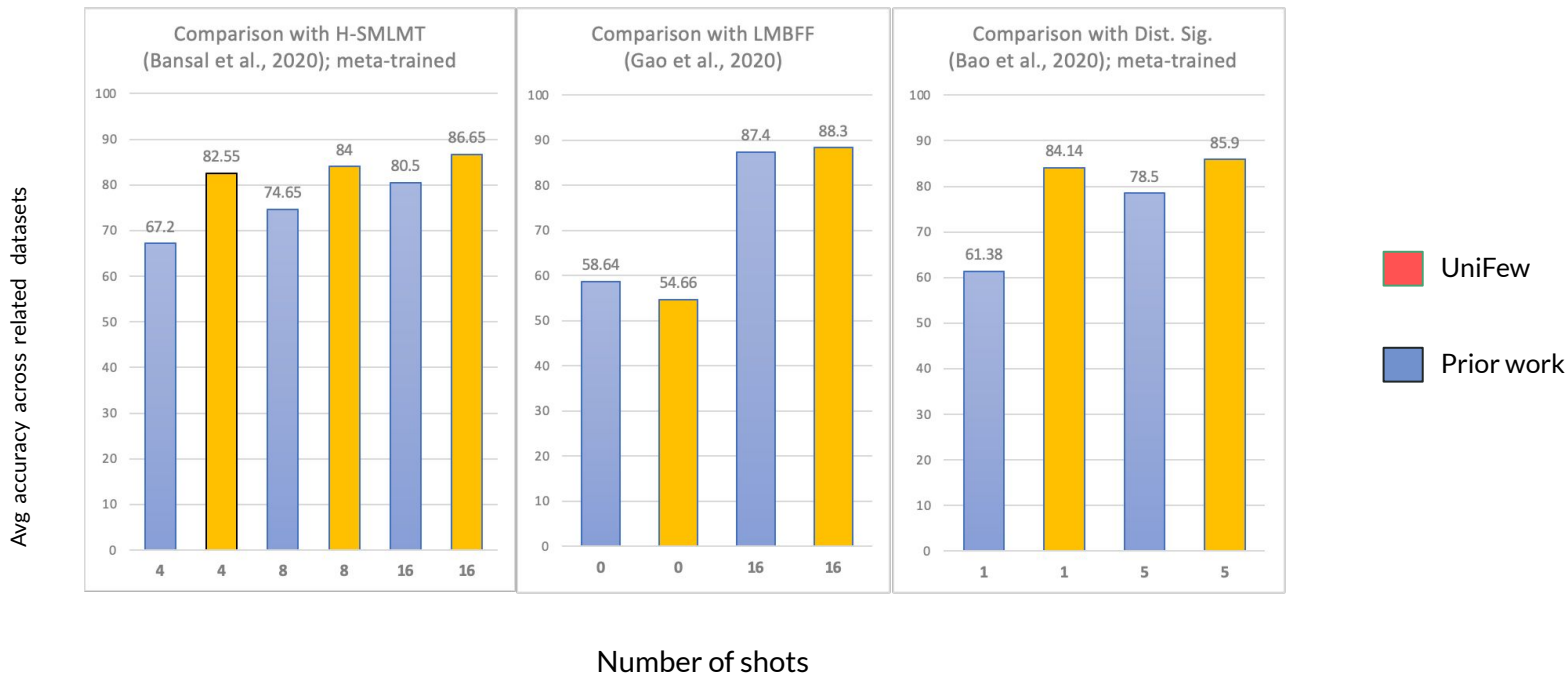
A T5-based model trained on a collection of QA datasets

Already works well for the Q/A task format and provides strong initialization

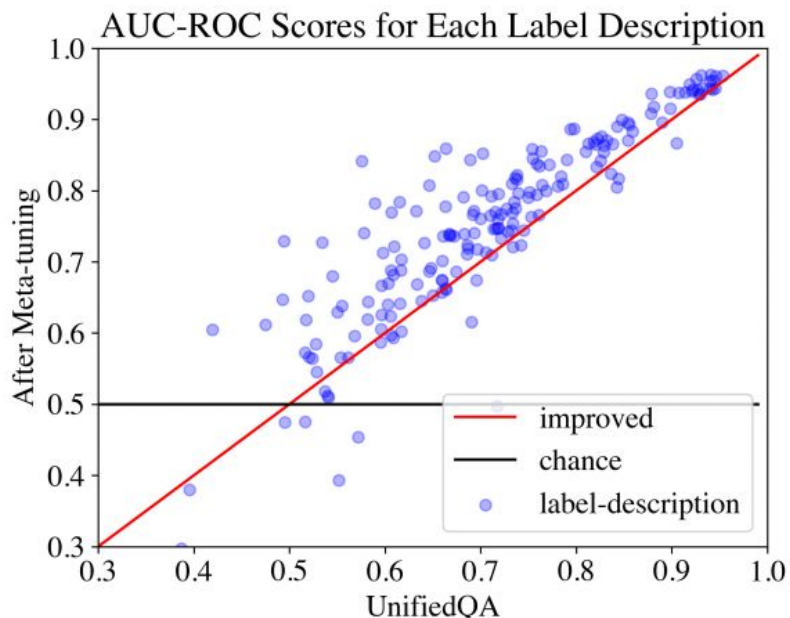
Meta-train on a suite of tasks from training set of FLEX and evaluate on unseen FLEX tasks.

Zhong et al., (2021) perform similar meta-training (they refer to it as "**meta-tuning**")

# UniFew



# Meta-tuning



Model performance  
improves after meta-tuning

$\Delta$  Meta-tuned vs. UnifiedQA = 3.3%



## Similar results in UniFew

	Zero-shot				Few-shot			
	Class	Domain	Task	Overall	Class	Domain	Task	Overall
UniFew	59.5	67.9	36.6	56.5	75.8	72.4	54.3	69.3
UniFew <sub>meta</sub>	75.6	87.6	41.1	71.0	80.2	86.8	62.4	77.9
$\Delta_{\text{meta}}$	+16.2	+19.7	+4.5	+14.5	+4.3	+14.4	+8.1	+8.6

UniFew results on FLEX. Performance breakdown based on different types of generalizations

# Meta-training - scaling up

Meta-training can be performed on large set of tasks and on larger models

Improved zero-shot performance and cross-task generalization

TO [\(Sanh et al., \(2022\)\)](#) and FLAN [\(Wei et al., \(2022\)\)](#)

# Scale is crucial for zero-shot

Problem: Zero-shot performance only gets well when the model size is very large  $> 100\text{B}$

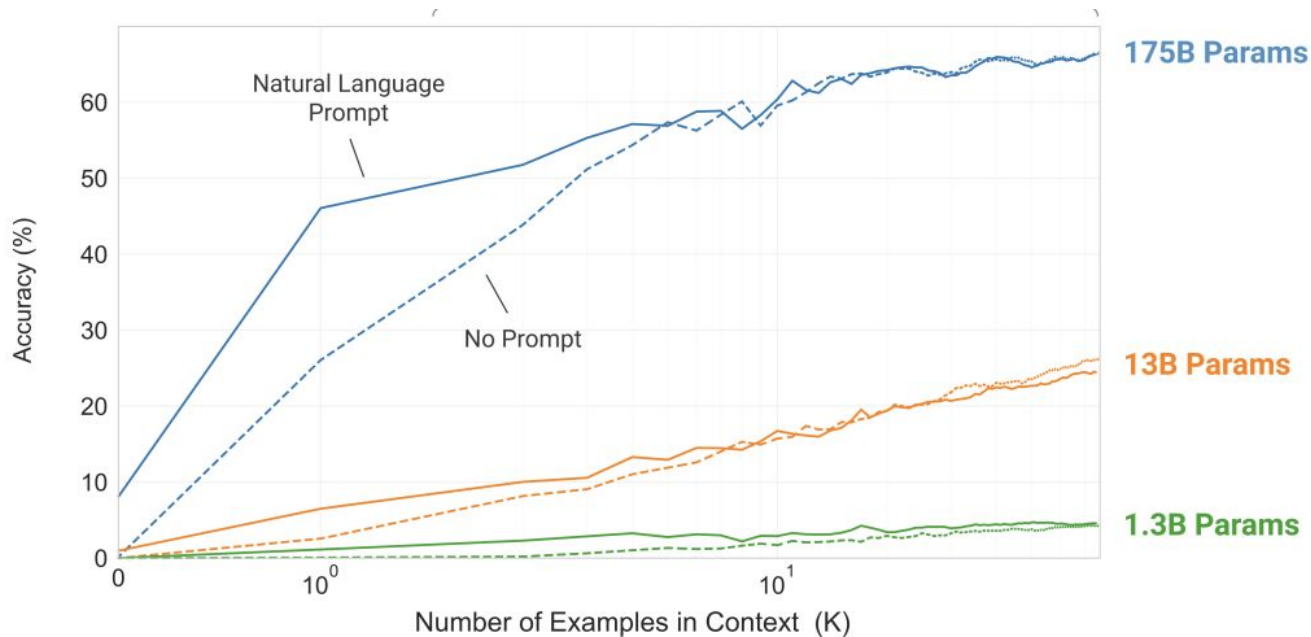


Fig from (Brown et al., 2020)

# Multitask Prompted Training

Why only large LMs work well for zero-shot?

Hypothesis: Implicit multi-task training during pre-training

**T0**: Explicitly multitask train with prompts

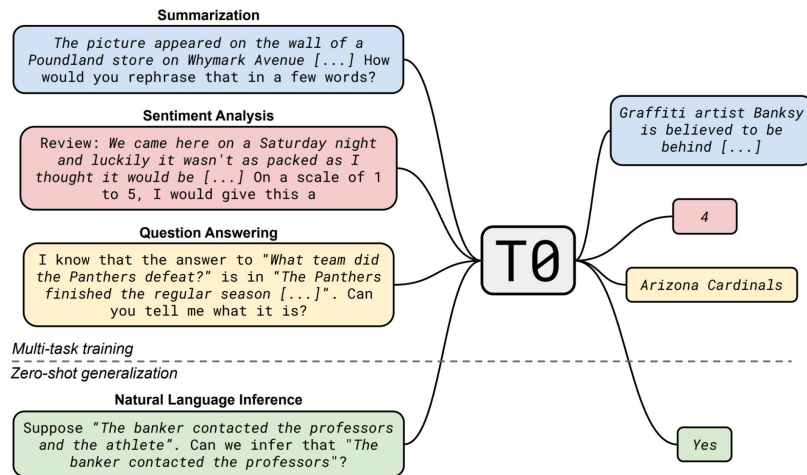
Feb 11, 2008

Hi everybody, I have a question.

Do these sentences mean the same?

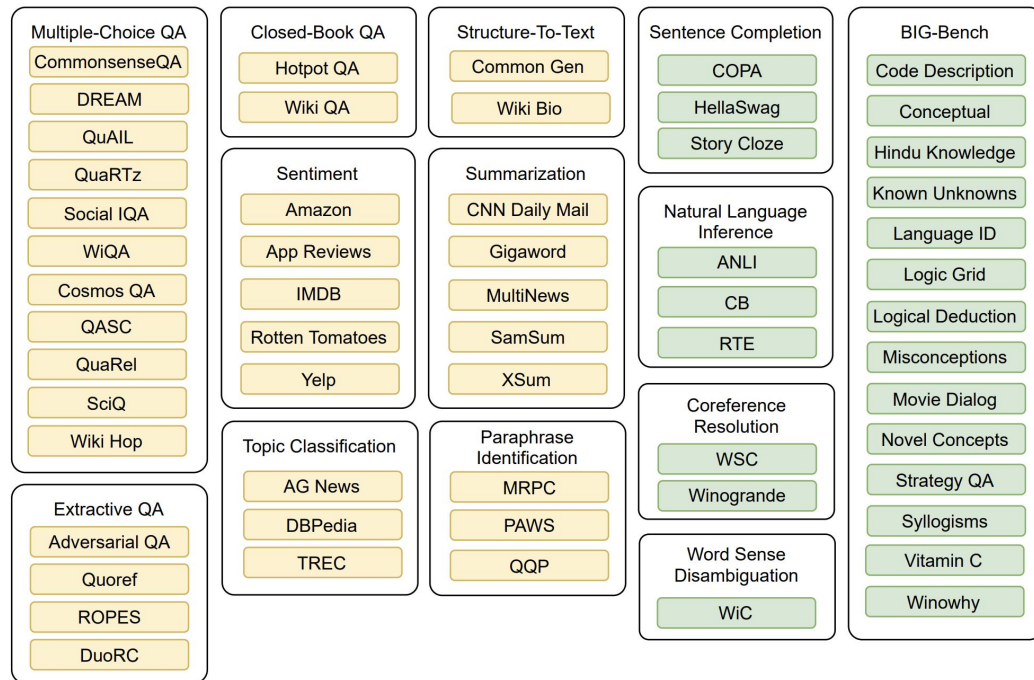
No other boy in this class is as smart as the boy.

No other boy is as smart as the boy in this class.



## T5 → T0

- Meta-train on a large mixture of NLP datasets
- Format examples of each dataset with multiple prompts
- Unified templated format
- Prompts collected through the BigScience workshop
- Interface: Promptsource
- 62 datasets, 520 prompts



Datasets used in T0

(Yellow used for training, green is held out)

# T0 - training details

Continue training T5 by combining and shuffling all examples from all training datasets

Base model: LM-adapted T5 (Lester et al., 2020), trained on 100B tokens from C4 on LM task.

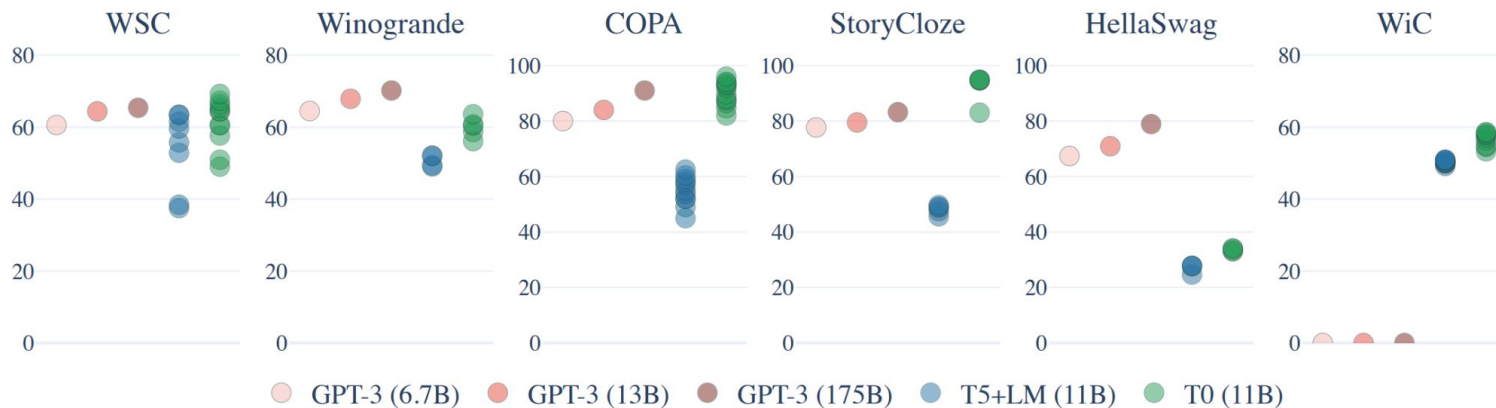
## Natural Language Inference



## Coreference Resolution

## Sentence Completion

## Word Sense

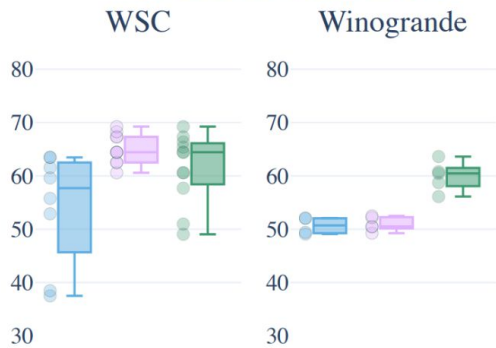


● GPT-3 (6.7B)
 ● GPT-3 (13B)
 ● GPT-3 (175B)
 ● T5+LM (11B)
 ● T0 (11B)

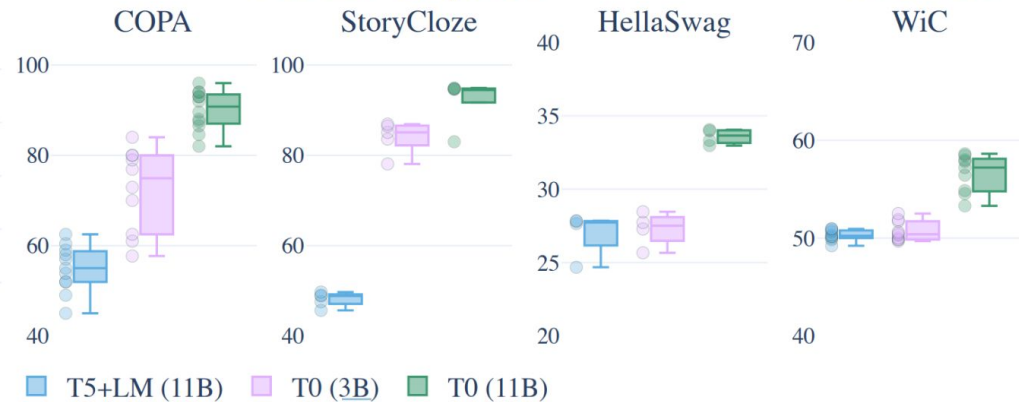
## Natural Language Inference



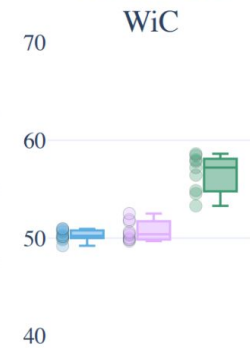
## Coreference Resolution



## Sentence Completion



## Word Sense



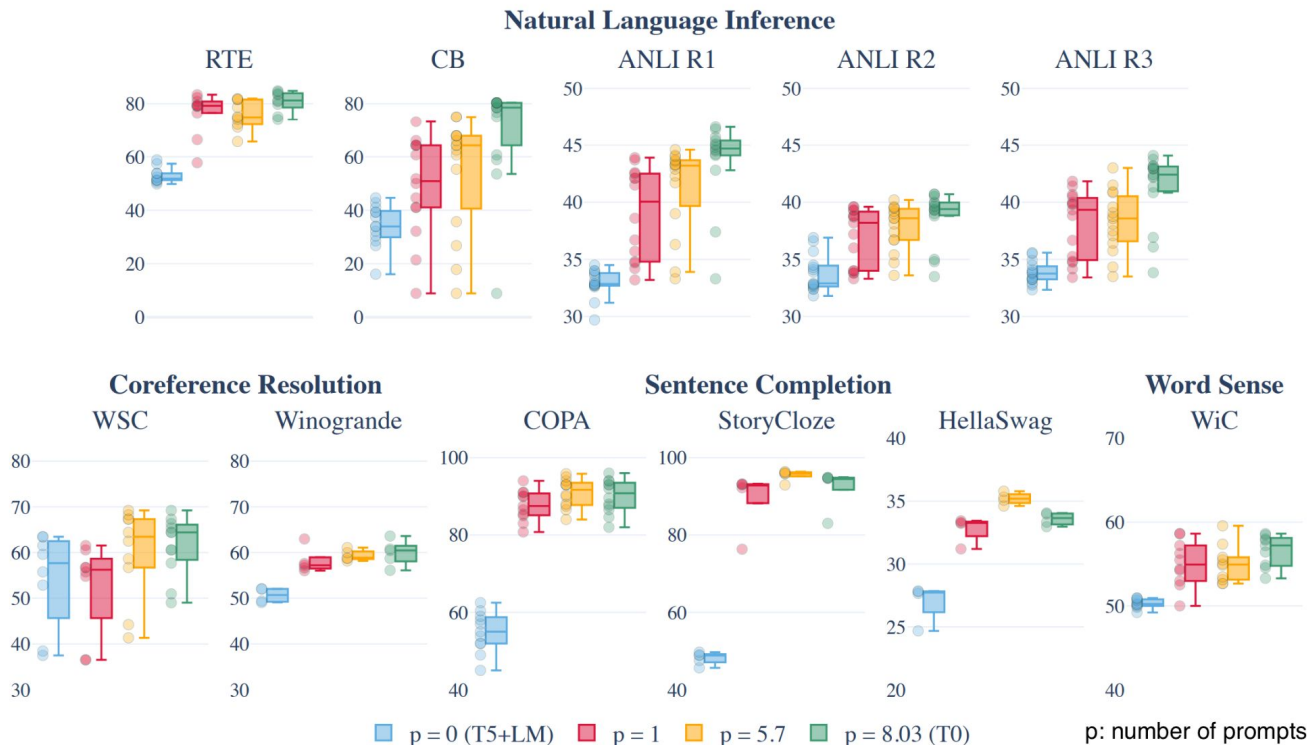
Legend: T5+LM (11B) (blue), T0 (3B) (purple), T0 (11B) (green)

It also works for smaller size models (T5 3B)



Trainin

ipts



Increasing the number of prompts improves the results and robustness

# FLAN: Finetuned Language Net

Very similar to T0

They refer to the meta-training stage as "instruction tuning"

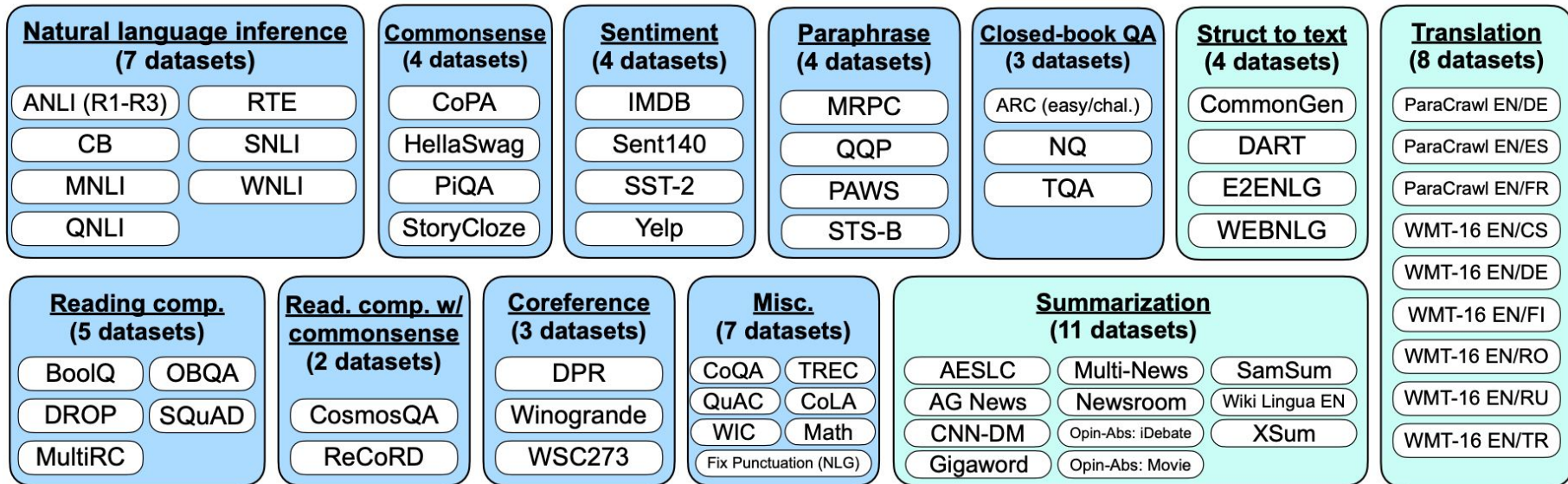
Finetuning LMs on a collection of datasets described via instructions

Compared with T0:

They use a larger model (137B vs 11B)

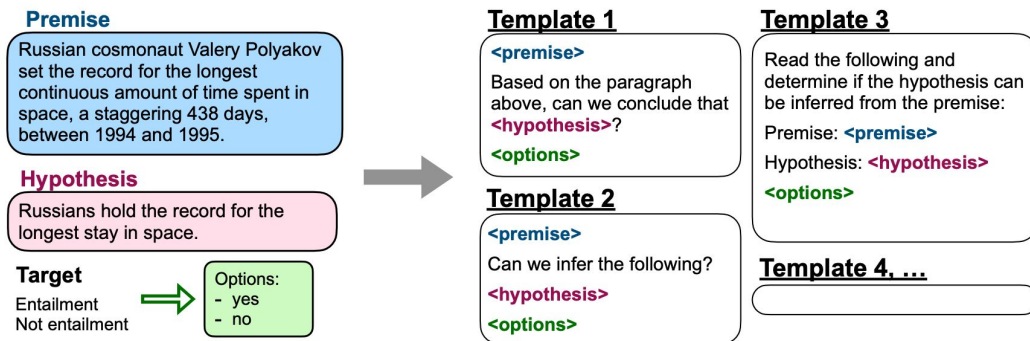
They use a decoder-only Transformer as the base model

# Tasks considered in FLAN



# FLAN

For each dataset write 10 templates



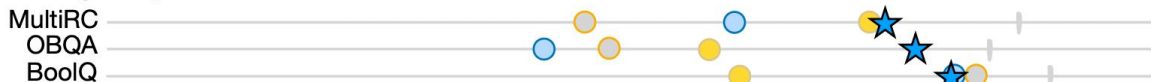
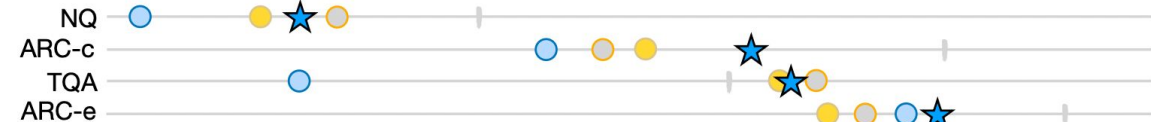
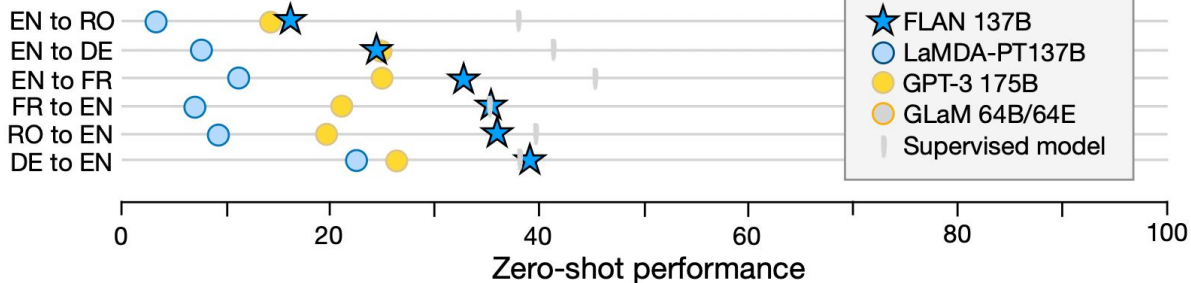
Similar to UniFew and T0, they add the valid options at the end of the prompt  
Compared with T0, prompts look more similar to instructions

Evaluation setup: Same as T0, hold out clusters of tasks

[Wei et al., \(2022\)](#) Finetuned Language Models Are Zero-Shot Learners

# FLAN

Base model: LaMDA-PT, a decoder-only LM of 137B params (Thoppilan et al., 2022).

**Natural language inference****Reading comprehension****Closed-book QA****Translation**

[Wei et al., \(2022\)](#) Finetuned Language Models Are Zero-Shot Learners

# Some tasks do not see benefits

Downstream tasks that are similar to LM pre-training objective instruction tuning is not useful. (coreference and commonsense)

Task	LAMDA-PT	FLAN
Winogrande	68.3	67.3
WSC273	81.0	80.8
COPA	90.0	90.6
HellaSwag	57.0	56.4
PIQA	80.3	80.9
StoryCloze	79.5	92.2

## INPUT

How does the sentence end?

Elena wanted to move out of her parents fast but Victoria wanted to stay for a while,

## OPTIONS:

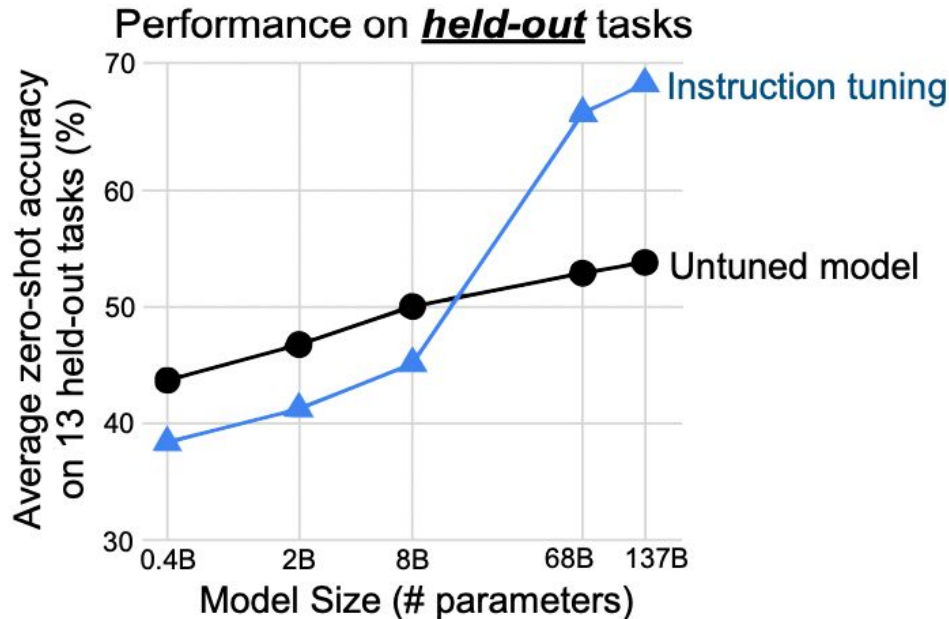
- Elena went to school.
- Victoria went to school.

## TARGET

Victoria went to school.

An example from Winogrande

# Scale



Their results show that instruction tuning **only works for large models > 8B**

However, T0 achieves strong results even with 3B model

Two potential factors:

- T0 uses encoder-decoder
- They claim that their prompts are more diverse

More work is required to understand the scale vs performance of meta-training



# Other objective functions used in meta-training

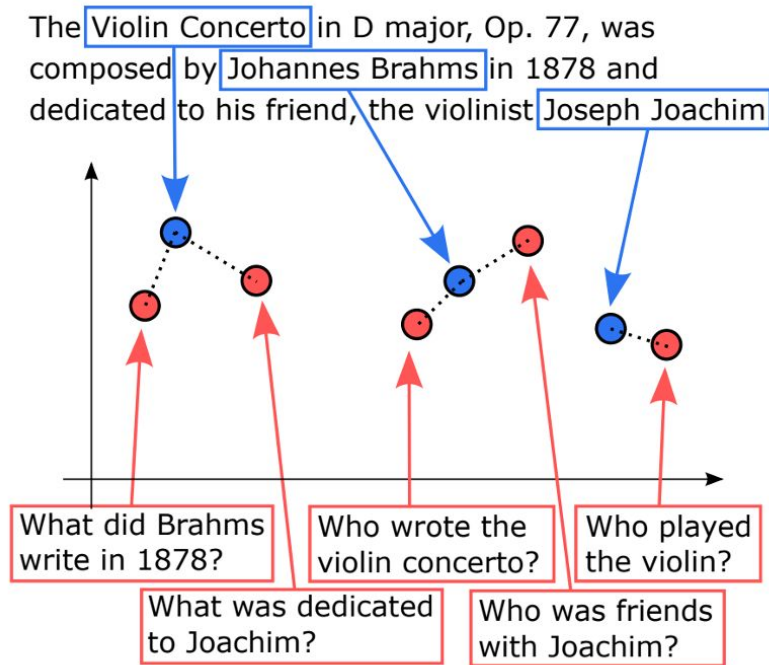
So far we've seen meta-training using unifying task formats and multitasking

Some works employ different types of objective functions for improving model performance

# Other objective functions for meta-training

## QA Infused

Contextual representation of a phrase should answer all questions about that phrase



# QA Infused pretraining

Train with a biencoder extractive QA objective

Use BART-large for question/answer generation.

The model independently encodes the question and passage

Apply dot product b/w passage representation at index  $i$  and the question to find the correct start/end index of the answers

# QA Infused pretraining

Model	QQP	MRPC	PAWS-Wiki	PAWS-QQP
RoBERTa-large	.763	.831	.698	.690
Cross-encoder + MRQA	.767	.840	.742	.731
QUIP, cross-encoder student	.769	.847	.751	.706
Bi-encoder + UnsupervisedQA	.747	.801	.649	.580
Bi-encoder + MRQA	.771	.807	.747	.725
QUIP, no teacher	.767	.831	.780	.709
QUIP	<b>.809</b>	<b>.849</b>	<b>.830</b>	<b>.796</b>

Zero-shot paraphrase ranking (BERTScore)

# QA Infused pretraining

Model	Fine-tuned?	QQP	MRPC	PAWS-Wiki	PAWS-QQP
LM-BFF (reported)	Fine-tuned	69.8 <sub>0.8</sub>	77.8 <sub>0.9</sub>	-	-
LM-BFF (rerun)	Fine-tuned	67.1 <sub>0.9</sub>	76.5 <sub>1.5</sub>	60.7 <sub>0.7</sub>	50.1 <sub>2.8</sub>
RoBERTa-large	Frozen	64.4 <sub>0.4</sub>	80.6 <sub>0.7</sub>	62.3 <sub>0.9</sub>	50.6 <sub>0.4</sub>
QUIP	Frozen	68.9 <sub>0.2</sub>	82.6 <sub>0.4</sub>	71.9 <sub>0.5</sub>	<b>63.0<sub>1.2</sub></b>
RoBERTa-large	Fine-tuned	64.9 <sub>0.7</sub>	84.4 <sub>0.3</sub>	65.7 <sub>0.3</sub>	50.9 <sub>0.8</sub>
QUIP	Fine-tuned	<b>71.0<sub>0.3</sub></b>	<b>86.6<sub>0.4</sub></b>	<b>75.1<sub>0.2</sub></b>	60.9 <sub>1.0</sub>

few-shot paraphrase classification

# Meta-training for In-Context Learning (ICL)

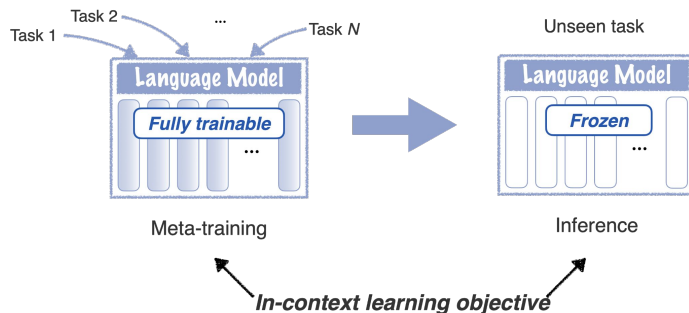
Advantages of ICL:

- No need for fine-tuning or training

- Easy adaptation to new tasks

However, their performance is not always as good as fine-tuning

“Can we meta-train an LM to be a better in-context learner?”



# MetaCL

## Meta-Training

- Given  $C$  meta-training tasks
- Sample a task  $T$
- Sample examples  $(x_1, y_1), \dots, (x_{k+1}, y_{k+1})$  from  $T$
- Last example acts as test
- Maximize  $P(y_{k+1} | x_1, y_1, \dots, x_k, y_k, x_{k+1})$



## Inference

- Given training examples  $(x_1, y_1), \dots, (x_k, y_k)$
- Find  $\operatorname{argmax}_c P(c | x_1, y_1, \dots, x_k, y_k, x)$

## Meta-Training

- Given  $C$  meta-training tasks
- Sample a task  $T$
- Sample examples  $(x_1, y_1), \dots, (x_{k+1}, y_{k+1})$  from  $T$
- Last example acts as test
- Maximize  $P(y_{k+1} | x_1, y_1, \dots, x_k, y_k, x_{k+1})$



## Inference

- Given training examples  $(x_1, y_1), \dots, (x_k, y_k)$
- Find  $\operatorname{argmax}_{y \in Y} P(y | x_1, y_1, \dots, x_k, y_k, x)$

## MetalCL

## Meta-Training

- Given  $C$  meta-training tasks
- Sample a task  $T$
- Sample examples  $(x_1, y_1), \dots, (x_{k+1}, y_{k+1})$  from  $T$
- Last example acts as test
- Maximize  $P(y_{k+1} | x_1, y_1, \dots, x_k, y_k, y_{k+1})$



## Inference

- Given training examples  $(x_1, y_1), \dots, (x_k, y_k)$
- Find  $\operatorname{argmax}_{y \in Y} P(x | x_1, y_1, \dots, x_k, y_k, y)$

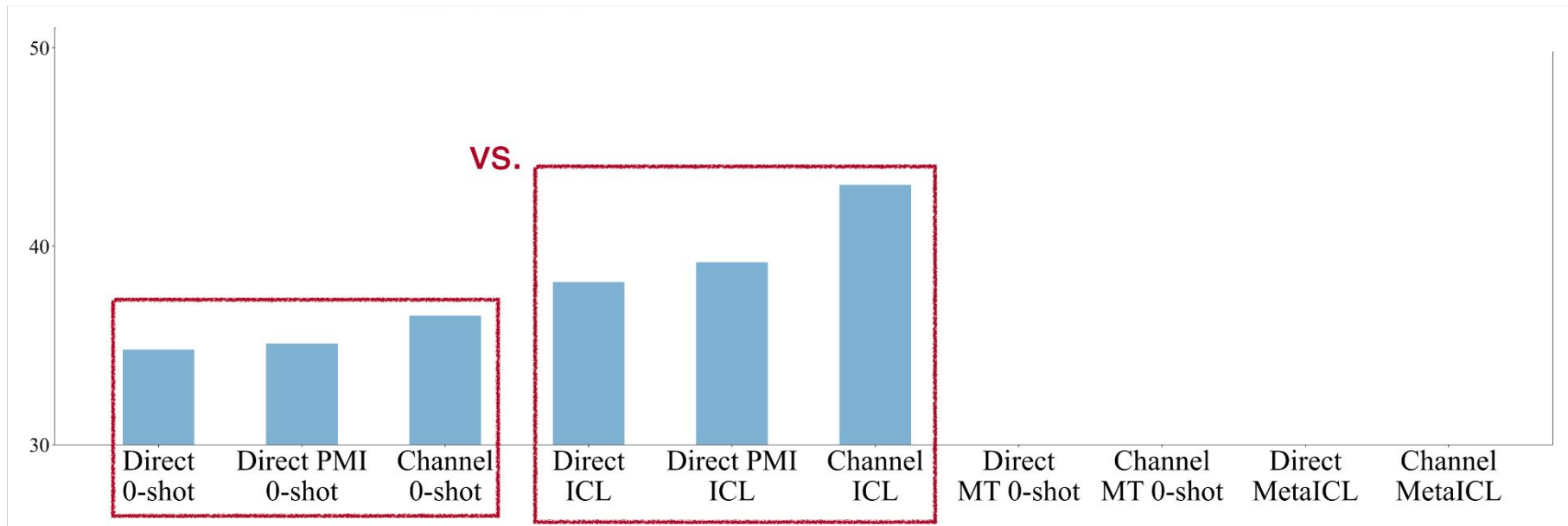
## Channel MetalCL



# Results

Base LM: GPT-2 Large

Raw LM



\* *Direct PMI is a better direct model from Holtzman et al, Zhao et al.*

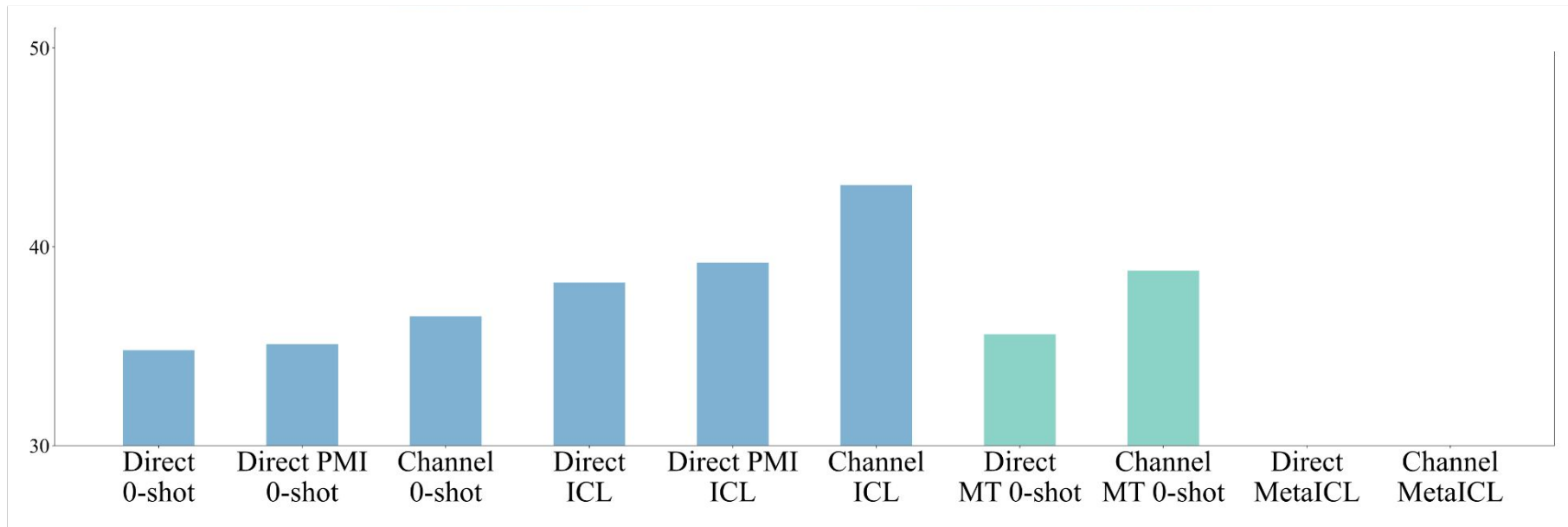
[Min et al \(2021\)](#) MetaICL: Learning to Learn In Context

# Results

Multi-task training  
but *w/o* in-context learning  
(Similar to FLAN and T0)

Raw LM

Multi-task 0-shot

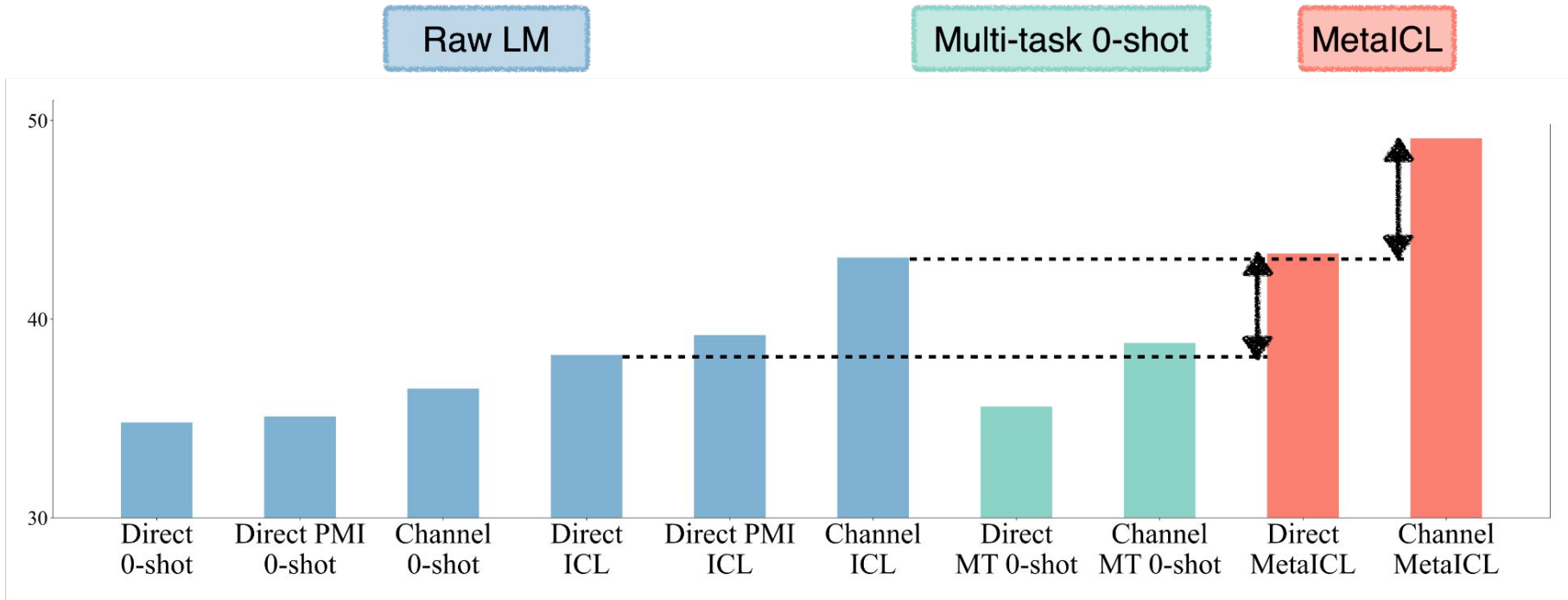


Multi-task 0-shot is not as competitive

[Min et al \(2021\)](#) MetaICL: Learning to Learn In Context

# Results

Base LM: GPT-2 Large



[Min et al \(2021\)](#) MetaICL: Learning to Learn In Context

# Additional self-supervised objectives to improve ICL

Using additional self-supervised objectives to improve ICL (Chen et al., 2022)

Self-supervised tasks:

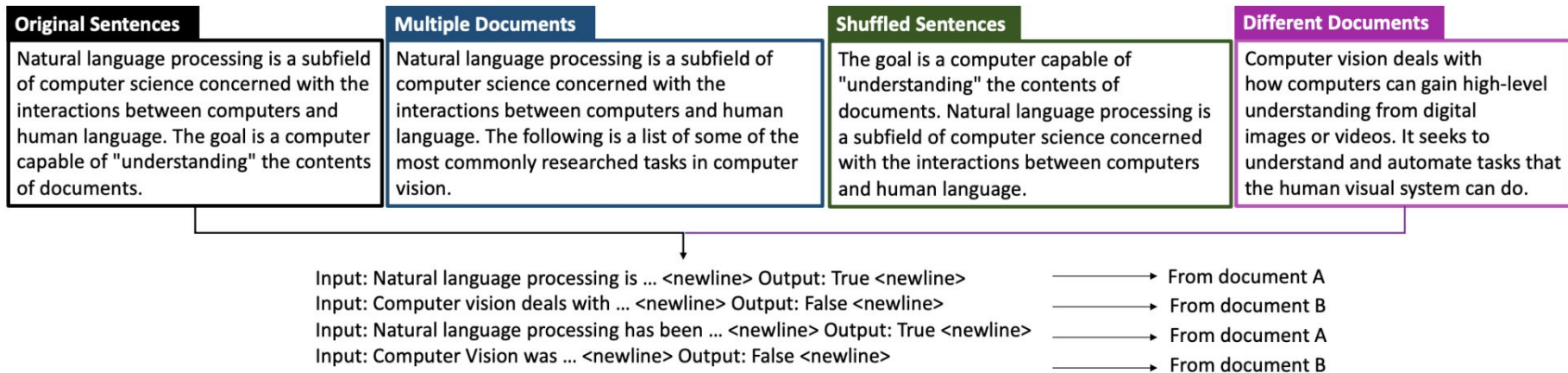
- Next Sentence Generation: Generate the last sentence of the doc
- Masked word prediction
- Last Phrase Prediction: Predict last phrase in a sentence
- Classification (similar to Next Sentence Prediction and Shuffling objectives)

**Question:** The goal is a computer capable of "understanding"? **Output:** the contents of documents.

**Question:** The goal is a computer capable of "understanding"? **Answer:** the development of new models. **Output:** False

# Additional self-supervised objectives to improve ICL

- Classification (similar to next sentence prediction or shuffling objectives)



# Additional self-supervised objectives to improve ICL

Results on few-shot tasks obtained from SuperGLUE dev set

Model	MS	Avg.
LM	125M	48.4
ExtraLM	125M	48.3
NewTemplate	125M	46.2
CrossTask(NLI→QA)	125M	42.2
CrossTask (QA→NLI)	125M	
SameTask	125M	61.9
Self-Supervised	125M	<b>51.0</b>

LM	1.3B	51.8
ExtraLM	1.3B	51.7
NewTemplate	1.3B	50.7
CrossTask(NLI→QA)	1.3B	49.6
CrossTask (QA→NLI)	1.3B	
SameTask	1.3B	69.9
Self-Supervised	1.3B	<b>55.6</b>

On average larger model achieves larger gains.

This is consistent with other reported results in the literature

[Chen et al.. \(2022\)](#) Improving In-Context Few-Shot Learning via Self-Supervised Training

# Additional self-supervised objectives to improve ICL

<b>Model</b>	<b>BQ</b>	<b>MC</b>	<b>CA</b>	<b>RE</b>	<b>CB</b>	<b>Avg.</b>
LM	52.2	26.6	63.0	50.8	38.3	46.2
ALL	55.7	33.6	67.6	53.0	44.9	51.0
ALL-NSG	55.0	31.8	62.7	52.5	45.9	49.6
ALL-MWP	55.6	33.5	67.3	52.7	45.5	50.9
ALL-LPP	53.5	30.5	67.6	51.9	46.6	50.0
ALL-CL	54.0	32.9	67.4	52.8	39.0	49.2

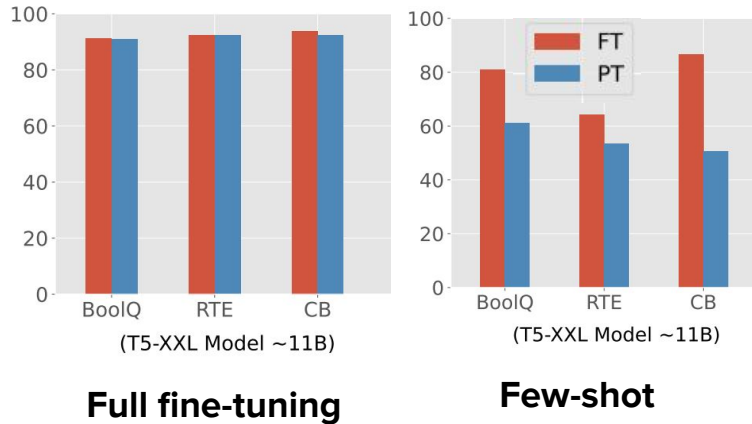
Ablation results on training objectives. NSG: Next sentence generation, MWP: Masked word prediction, LPP: last phrase prediction, CL: classification

# Parameter efficient fine-tuning for few-shot learning

Can we meta-train param efficient models for better initialization?

Recall part 3 and parameter efficient fine-tuning methods.

Some of these approaches don't immediately work for few-shot learning





# Pretrained prompt tuning

A number of papers have explored the idea of pretraining param efficient models

In case of prompt tuning naive initialization from word embeddings corresponding to manual prompts doesn't help

	SST-2	BoolQ
Random Init.	70.5 <sub>15.5</sub>	61.0 <sub>5.3</sub>
Label Init.	58.9 <sub>2.7</sub>	63.0 <sub>0.4</sub>
Vocab Sampling	57.0 <sub>4.0</sub>	58.4 <sub>4.9</sub>
Top-1000 Sampling	57.9 <sub>4.2</sub>	57.7 <sub>3.9</sub>
Task-Related Sampling	58.5 <sub>3.8</sub>	58.2 <sub>4.0</sub>
<b>Full-Model Tuning</b>	<b>91.4<sub>0.8</sub></b>	<b>80.8<sub>2.4</sub></b>

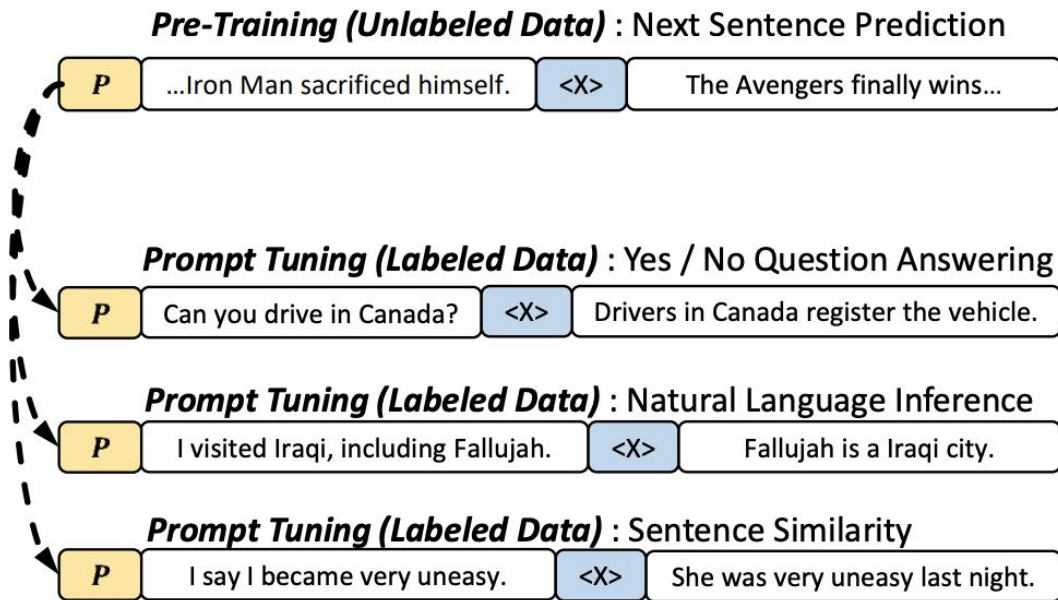
# Pre-trained prompt tuning

The key idea of PPT is pretraining with soft prompts to get better initializations

They introduce self-supervised tasks to pre-train the soft prompts

# Pre-trained Prompt Tuning

Use Next Sentence Prediction to pretrain with soft prompts on unlabeled data



# Pretrained Prompt Tuning

Use the CLS token for classification

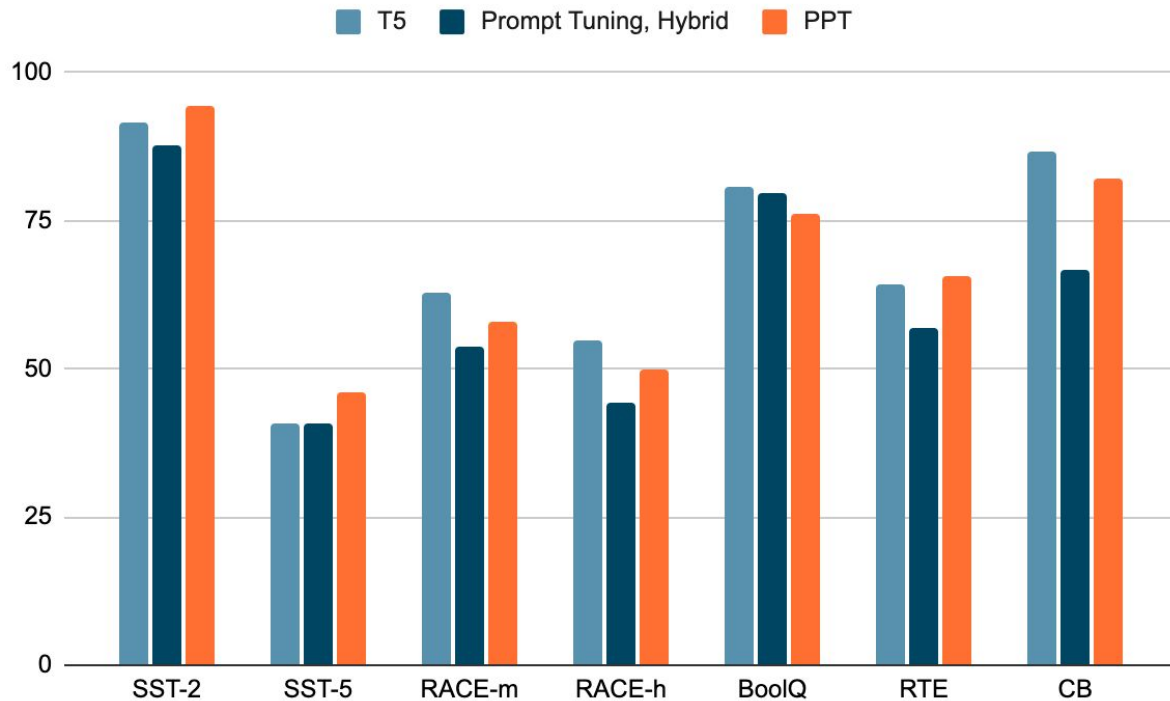
Tasks used for pretraining:

**NLI:** Label: (2) next sentence (1) from same document (0) random sentence

**Classification:** Given a sentence, choose adjacent sentence from 6 candidates

**Sentence classification:** Use another RoBERTa-based models to label sentences for sentiments

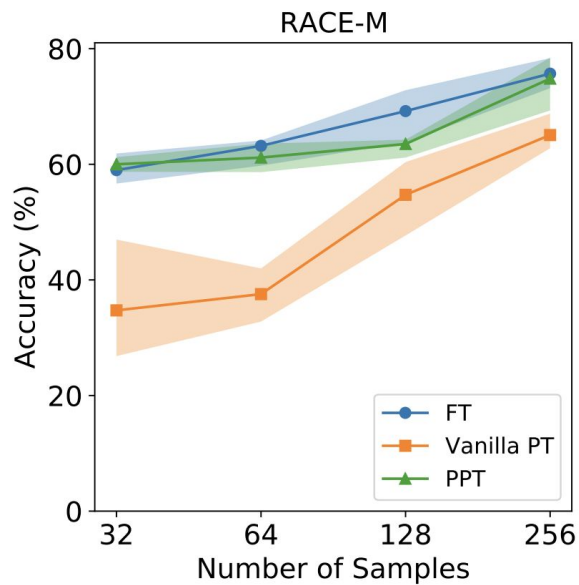
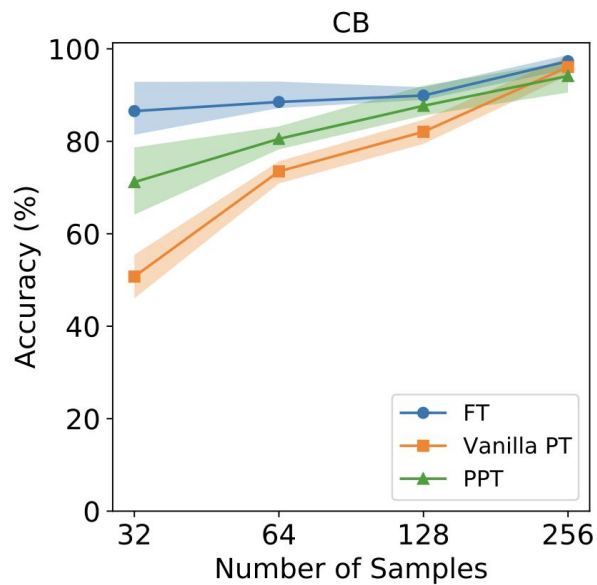
# PPT - Results



[Gu et al \(2021\)](#) PPT: Pre-trained Prompt Tuning for Few-shot Learning

# PPT

The pretraining of soft prompts helps most when there is less training data available



# Parameter efficient fine-tuning compared with ICL for FSL

ICL has some disadvantages:

- process all prompted input-target pairs every time we make a prediction

  - Possible to use caching but storage costs can be significant

- Performance is not better than fine-tuning

- Exact formatting of the prompt and order of examples matters

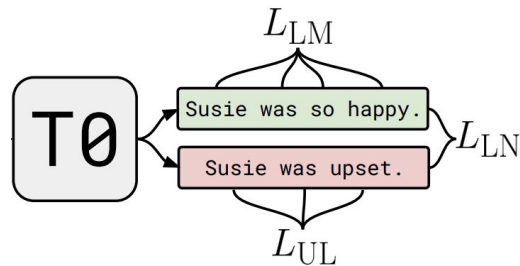
- Other work shows that even with wrong labels, ICL works

T-Few [Liu et al \(2022\)](#): A recipe for making **parameter efficient fine-tuning** work for few-shot learning

[Liu et al \(2022\)](#) Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning

# T-Few recipe

- **T0** as backbone
- **(IA)<sup>3</sup>** for downstream task adaptation





# Recall (IA)<sup>3</sup>

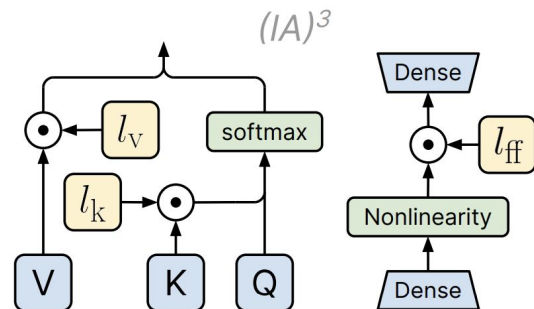
A parameter efficient fine-tuning method

Introduce rescaling to:

keys and values in self-attention

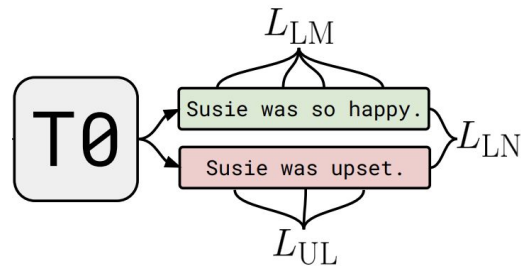
Keys and values in encoder-decoder attention

Intermediate activation of the position-wise  
feed-forward networks



# T-Few recipe

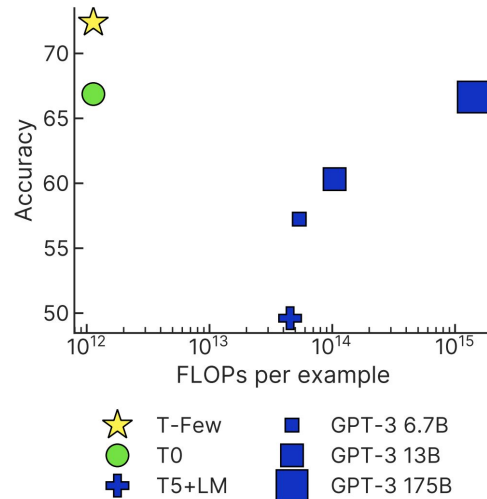
- **T0** as backbone
- **(IA)<sup>3</sup>** for downstream task adaptation
- **Meta-train (IA)<sup>3</sup>** on the multitask mixture of T0
- **3 Losses:** Language modeling (LLM), an unlikelihood loss LUL, and length normalized loss LLN.
- **Fixed hyperparams for all datasets:** Train for 1,000 steps with a batch size of 8 sequences using the Adafactor with LR=3e-3 and a linear decay schedule with a warmup of 60 steps.
- **Apply prompt templates** to downstream datasets during training and inference to convert each example into an instructive text-to-text format.



# T-Few compared with ICL

Method	Inference FLOPs	Training FLOPs	Disk space	Accuracy
T-Few	1.1e12	2.7e16	4.2 MB	72.4%
T0 [1]	1.1e12	0	0 B	66.9%
T5+LM [14]	4.5e13	0	16 kB	49.6%
GPT-3 6.7B [4]	5.4e13	0	16 kB	57.2%
GPT-3 13B [4]	1.0e14	0	16 kB	60.3%
GPT-3 175B [4]	1.4e15	0	16 kB	66.6%

Accuracy on held-out T0 tasks and computational costs for different few-shot learning methods and models.



# Summary

Meta-training is effective for improving zero-shot, few-shot, and cross task generalization

Key ideas: use unified task formats, leverage templates and continue training the model on a mixture of meta-training tasks

Scaling up mixture of tasks and model sizes help improving results

Additional self-supervised tasks can improve both ICL and fine-tuning methods

With param efficient training, and additional loss functions, it is possible to achieve a single recipe for few-shot fine-tuning that outperforms ICL

# Open questions and future work

What type of information the models pick up during meta-training? How such information is different from pretraining?

Some results are inconsistent across works. The choice of model architecture, pre-training and meta-training data, format of task templates, etc conflates numbers. More about model architecture in next part.

# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation benchmark [Arman]
- 17:05–17:25 Part 6: Meta-training [Arman]
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]

# Part 7: Pretraining considerations for zero/few-shot learning

---

Slides available at:

<https://github.com/allenai/acl2022-zerofewshot-tutorial>

# Overview

Previous sections: adapting an **existing** pretrained model to a new task.

This section: considerations for how to **build** these models to improve **zero/few-shot** performance

## Considerations

- Scaling
- Architecture and Objective
- Dataset
- Efficient pretraining

Remember - this is a brief overview, not a comprehensive literature review



# Scaling

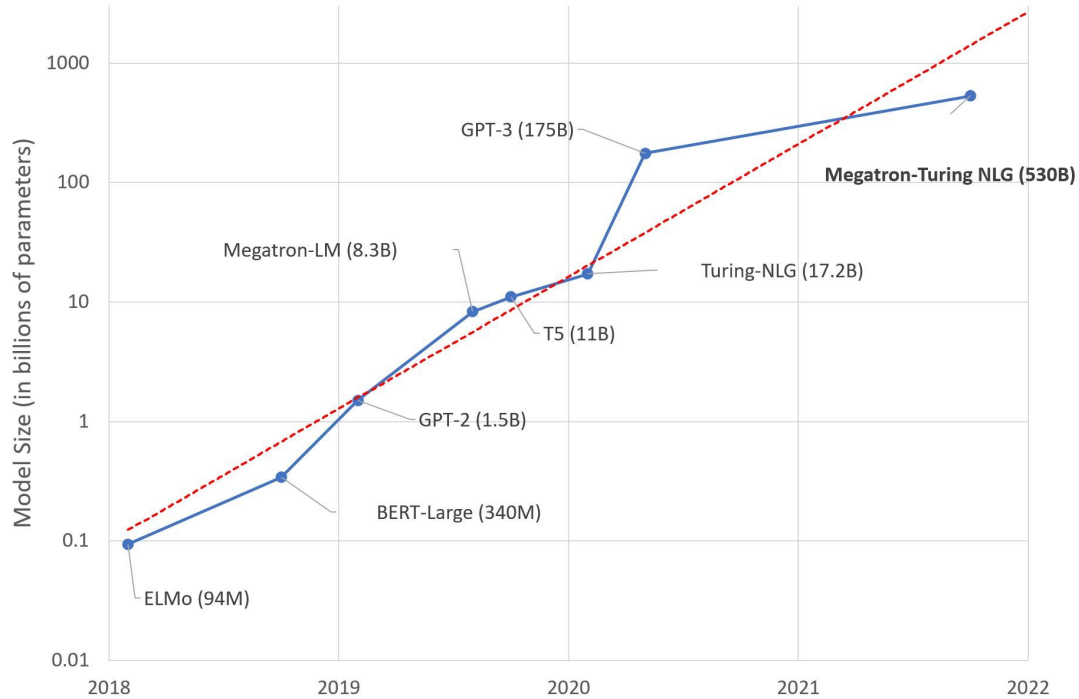


Photo credit: Microsoft Research Blog, Alvi et. al., 2021

LM are getting  
larger and more  
expensive

# Scaling

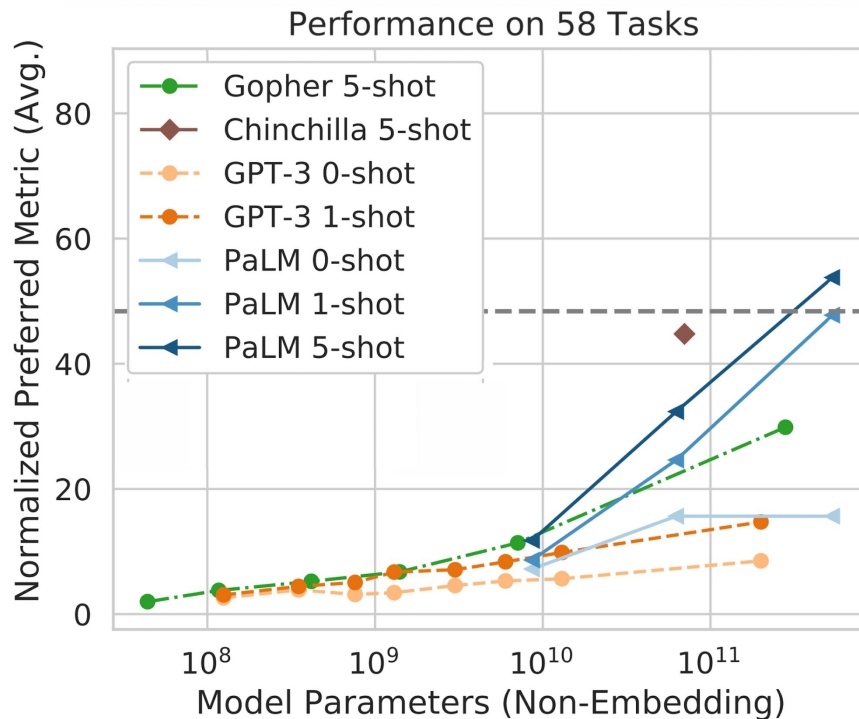


Photo credit: PaLM, Chowdhery et. al., 2022

Larger LMs  $\Rightarrow$   
better  
zero/few-shot  
performance

# Scaling

Misconception - “scaling means larger **model size**”

Correct - “scaling means more **compute**”

But - the extra compute is only useful if the model has enough capacity  $\Rightarrow$  a larger model

# Scaling

<b>Name</b>	<b>Size</b>	<b>Tokens</b>	<b>Compute (in GPT3-unit)</b>
GPT-NeoX	20B	472B	0.18x
GPT3	175B	300B	1x
Gopher	280B	300B	1.6x
Chinchilla	67B	1,400B	1.6x
Megatron-Turing-NLG	530B	270B	2.7x
PaLM	540B	780B	8x

# Scaling

<b>Name</b>	<b>Size</b>	<b>Tokens</b>	<b>Compute (in GPT3-unit)</b>
GPT3	175B	300B	1x
Gopher	280B	300B	1.6x
Chinchilla	67B	1,400B	1.6x
Megatron-Turing-NLG	530B	270B	2.7x
PaLM	540B	780B	8x

# Scaling - Optimal Model Size

Smaller models don't have enough capacity to utilize the extra compute. They plateau early.

Larger models are initially slower to train, but with more compute they reach lower losses.

Given a compute budget, what is the optimal model size to use?

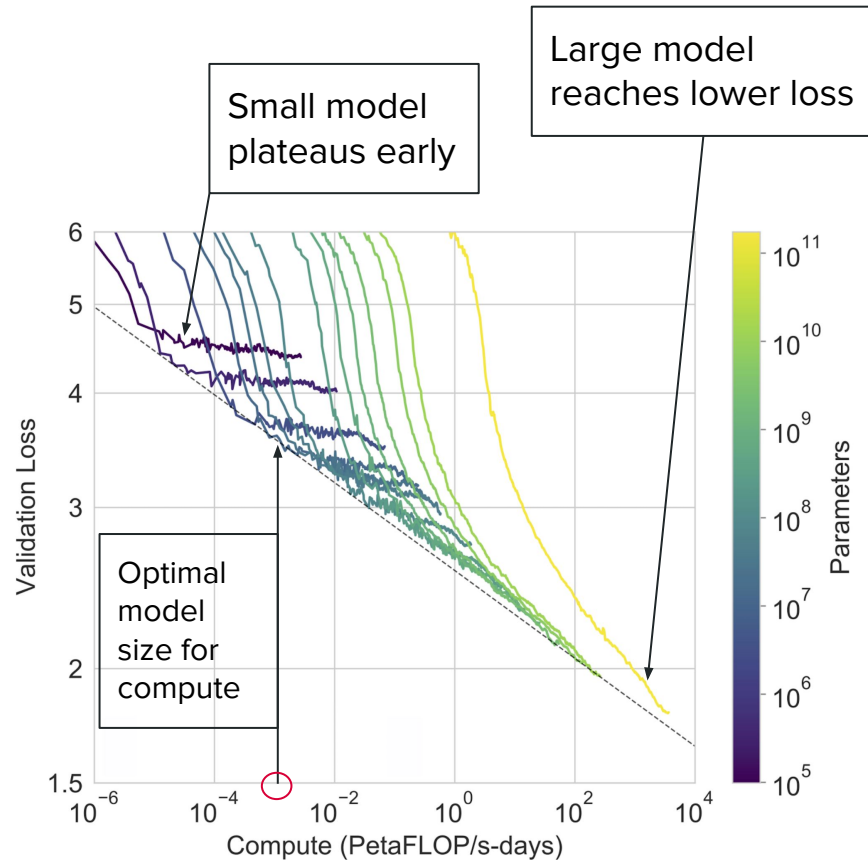


Photo credit: GPT3, Brown et. al., 2020

# Scaling - Kaplan et. al., 2020

The idea of “optimal model size for given compute” was introduced by Kaplan et. al.,

Train models to **optimality**, a lot earlier than **convergence**.

But we need to train the model to know its learning curve!! Not feasible when you have budget to train a single model.

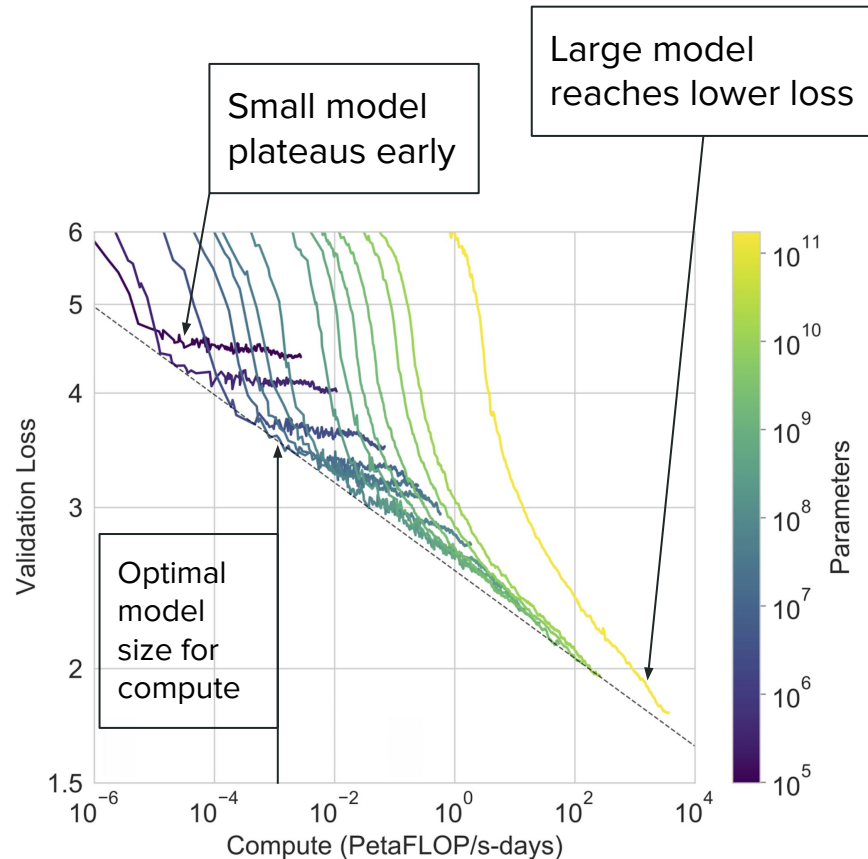


Photo credit: GPT3, Brown et. al., 2020

# Scaling - Kaplan et. al., 2020

**Scaling-laws:** learning curves are **power-law** functions of model size and compute.

Fit constants of the power-law function using small models, then extrapolate to large sizes.

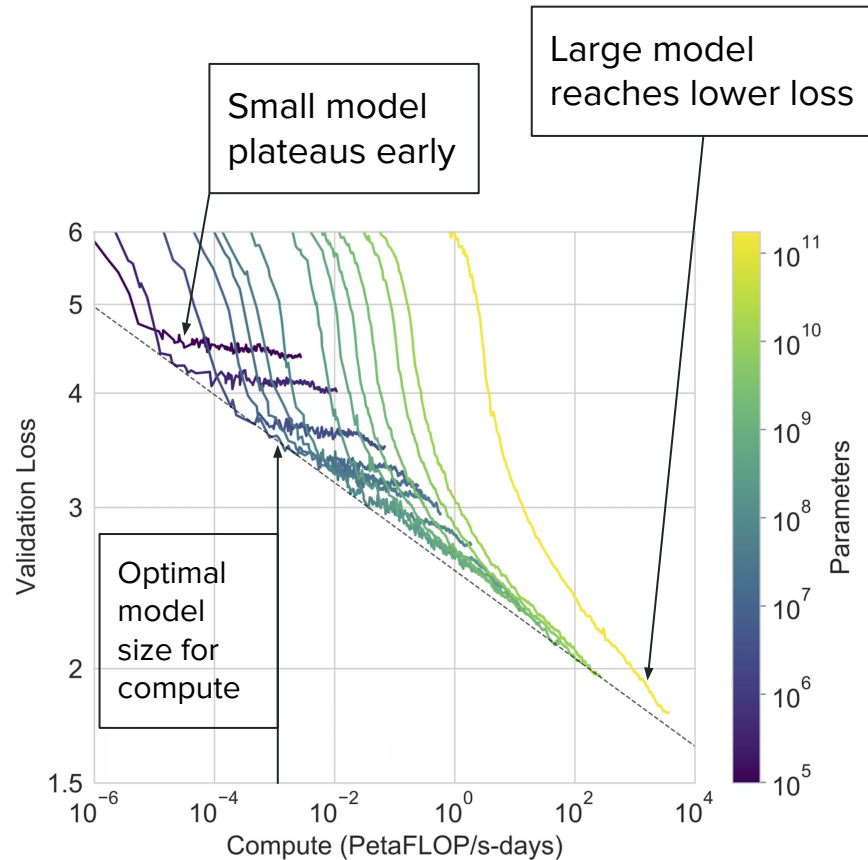


Photo credit: GPT3, Brown et. al., 2020



# Scaling - Kaplan et. al., 2020

The diagram illustrates the scaling law  $L(N, S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}(S)}\right)^{\alpha_S}$ . A box labeled "Constants" at the top has arrows pointing to the constants  $N_c$ ,  $S_c$ ,  $\alpha_N$ , and  $\alpha_S$  in the equation. A box labeled "Model size" at the bottom left has an arrow pointing to the variable  $N$  in the denominator of the first term. A box labeled "Number of steps" at the bottom right has an arrow pointing to the variable  $S_{\min}(S)$  in the denominator of the second term.

$$L(N, S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}(S)}\right)^{\alpha_S}$$

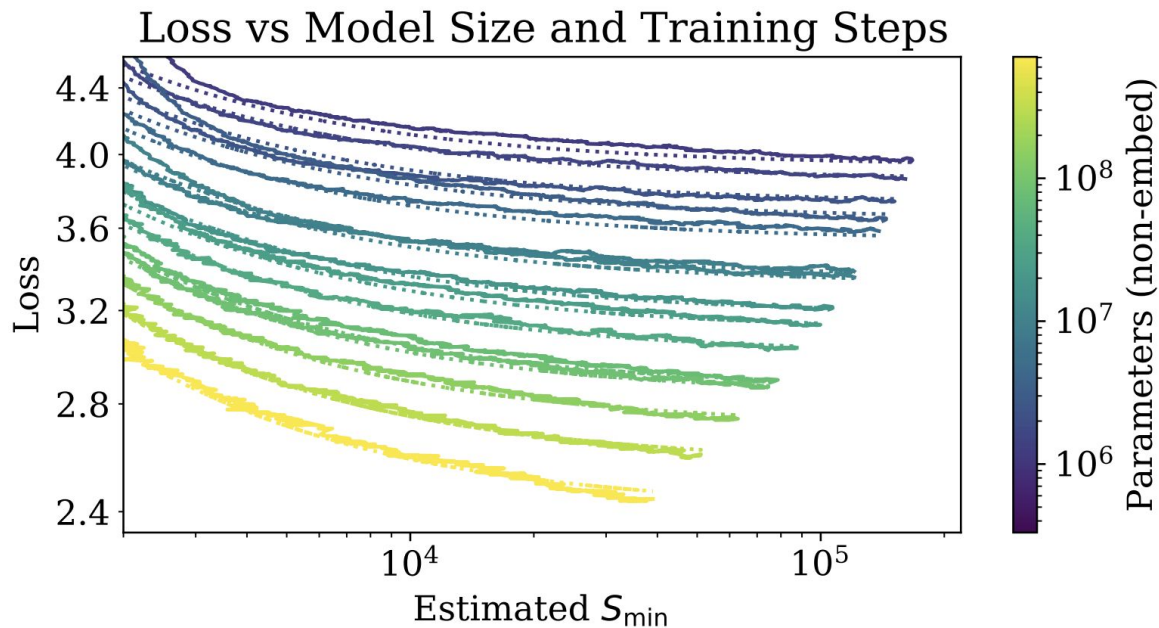
Model size

Number of steps

# Scaling - Kaplan et. al., 2020

Predicted (dotted) accurately matches empirical learning curves (solid).

Fit constants using small models then **predict** learning curves of larger models.



# Scaling - Kaplan et. al., 2020

Also, fit a power-law predicting the “compute efficient” frontier:  $L \propto C^{-0.048}$

Use to predict the optimal model size for a given amount of compute:  $N_{\text{opt}} \propto C^{0.73}$

and optimal number of tokens:  $D_{\text{opt}} \propto C^{0.27}$

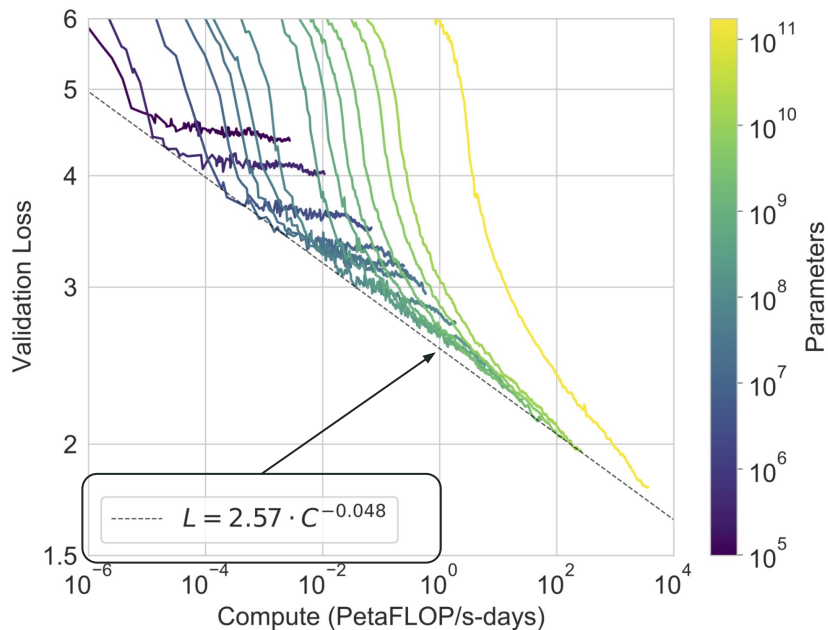


Photo credit: GPT3, Brown et. al., 2020

# Scaling - Kaplan et. al., 2020

Notations:

$$C = 6 N B S = 6 N D$$

C: compute

N: model size

B: batch size

S: number of steps

D: number of tokens (a single epoch, no repeated tokens)

# Scaling - Kaplan et. al., 2020

With more compute, should you increase model size or number of tokens?

$$C = 6 N D$$

C: compute

N: model size

D: number of tokens

# Scaling - Kaplan et. al., 2020

With more compute, should you increase model size or number of tokens?

Kaplan et. al.,	$N_{\text{opt}} \propto C^{0.73}$	$D_{\text{opt}} \propto C^{0.27}$
-----------------	-----------------------------------	-----------------------------------

$$N_{\text{opt}} \text{ exponent} \gg D_{\text{opt}} \text{ exponent}$$

Use the extra compute to grow the model size **faster** than growing number of tokens.

Given 10x compute, increase N by 5.5x, and D by 1.8x

GPT3 (and many other followed this recipe) training a 175B model on 300B tokens

# Scaling - Kaplan et. al., 2020 vs. Hoffmann et. al., 2022

With more compute, should you increase model size or number of tokens?

Kaplan et. al.,	$N_{\text{opt}} \propto C^{0.73}$	$D_{\text{opt}} \propto C^{0.27}$
Hoffmann et. al.,	$N_{\text{opt}} \propto C^{0.50}$	$D_{\text{opt}} \propto C^{0.50}$

$$N_{\text{opt}} \text{ exponent} \cong D_{\text{opt}} \text{ exponent}$$

Compute and tokens should increase **at the same rate**.

Given 10x compute, grow  $N$  by 3.2x and  $D$  by 3.2x

Hoffmann et. al., followed this recipe: Trained a 70B model on 1.4T tokens (Chinchilla) outperforming their previous 280B model trained on 300B tokens (Gopher)

# Scaling - Kaplan et. al., 2020 vs. Hoffmann et. al., 2022

The main difference is the learning rate schedule

Kaplan et. al.: all experiments for changing  $D$  used the same LR schedule with decay over  $2.5 \times 10^5$  steps

Hoffmann et. al.: each experiment changing  $D$  used a different LR schedule that reaches LR=zero at the end of training

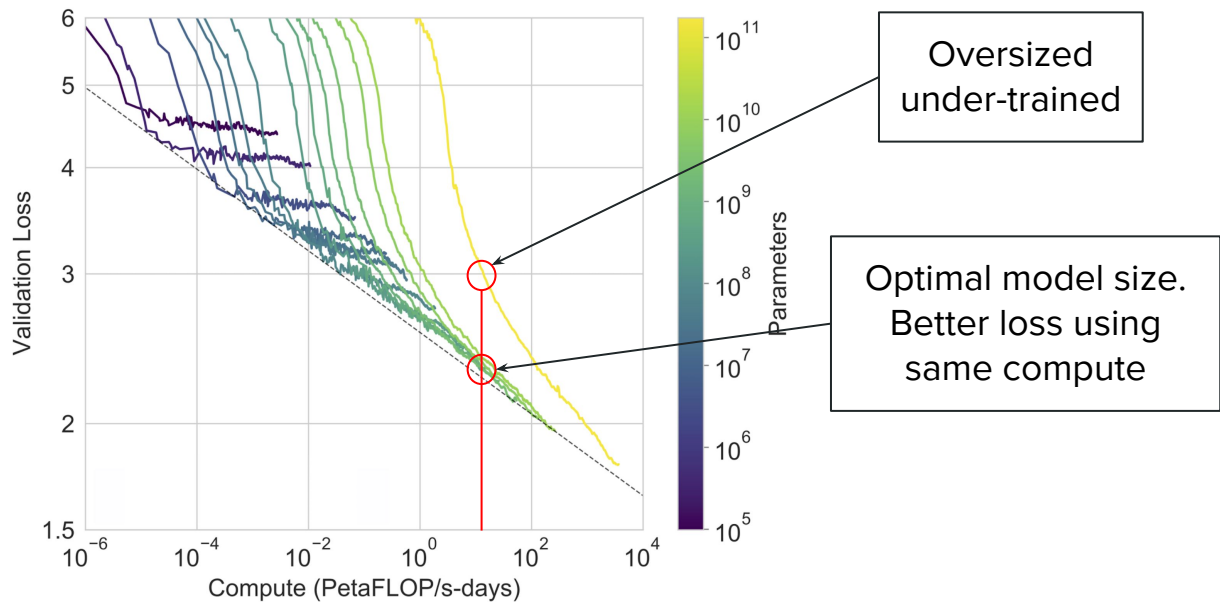
- The lower learning rate resulted in better loss, thus different empirical fit



# Scaling - Hoffmann et. al., 2022

Implications of Hoffmann et. al.,

- We trained many “oversized” and “under-trained” models



# Scaling - Hoffmann et. al., 2022

Implications of Hoffmann et. al.,

- We trained many “oversized” and “under-trained” models
- On the positive side, we already figured out the engineering for scaling to sizes much larger than needed
  - For example, optimality of 175B model (GPT3-size) at 3.7T tokens (12x more compute than GPT3)
- We should focus on increasing compute budget and data size not model size

# Overview

Previous sections: adapting an **existing** pretrained model to a new task.

This section: considerations for how to **build** these models to improve zero/few-shot performance

## Considerations

- Scaling
- **Architecture and Objective**
- Dataset
- Efficient pretraining

# Architecture and Pretraining Objective

Architecture and pretraining objectives have a big impact on the zero/few-shot results.

How well difference modeling choices work for different applications?

And do we have a universally good model design?

We will discuss:

- Survey of architectures
- Survey of pretraining objective
- Caveats and Challenges with model design research

# Architecture and Pretraining Objective

Typically, certain transformer architecture go well with certain objectives  
However, they can be disentangled. Most architectures can support most objectives.

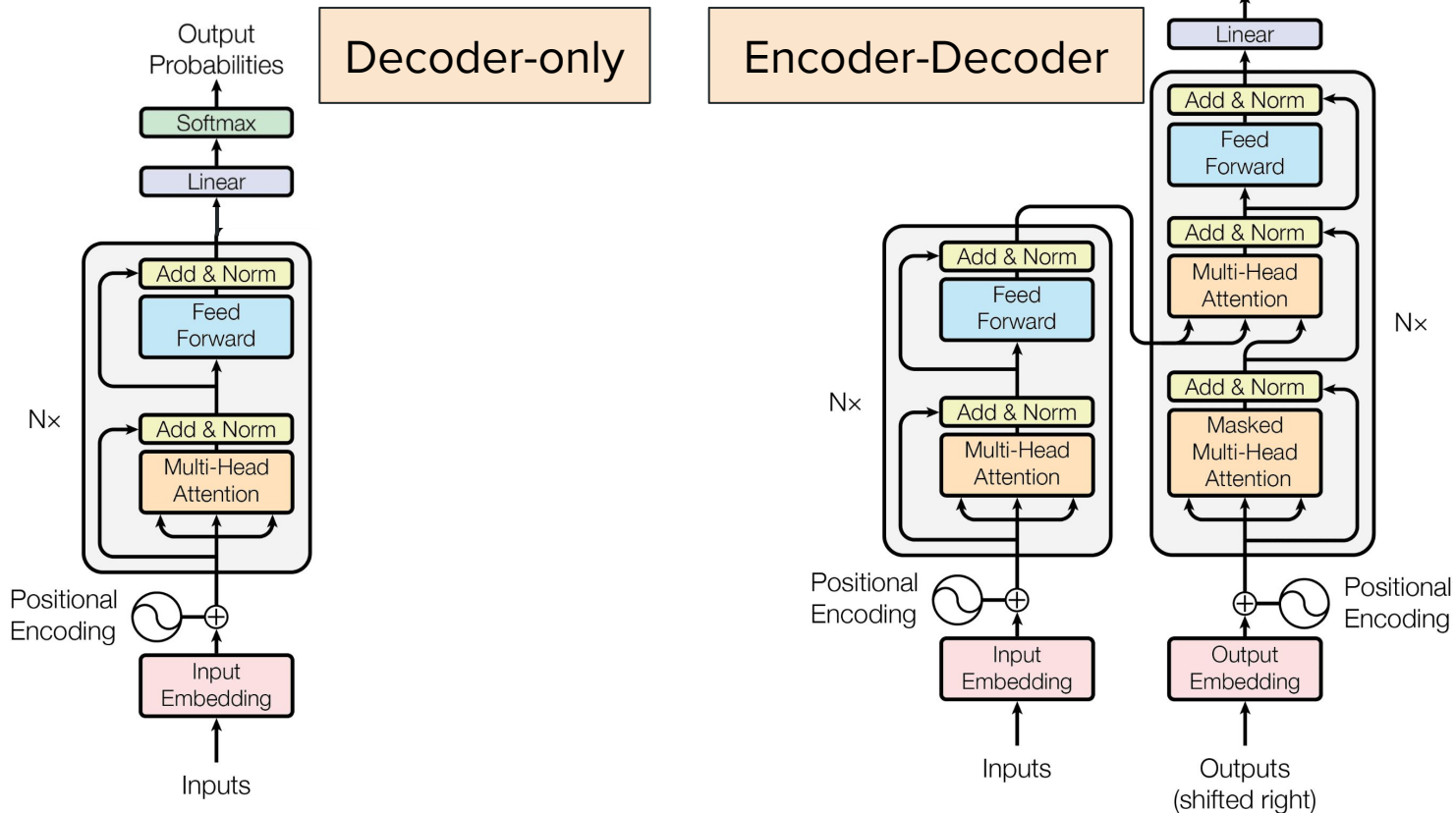
## Architectures:

- Vanilla transformer
  - Decoder-only (same architecture as encoder-only)
  - Encoder-decoder
- Transformer variants

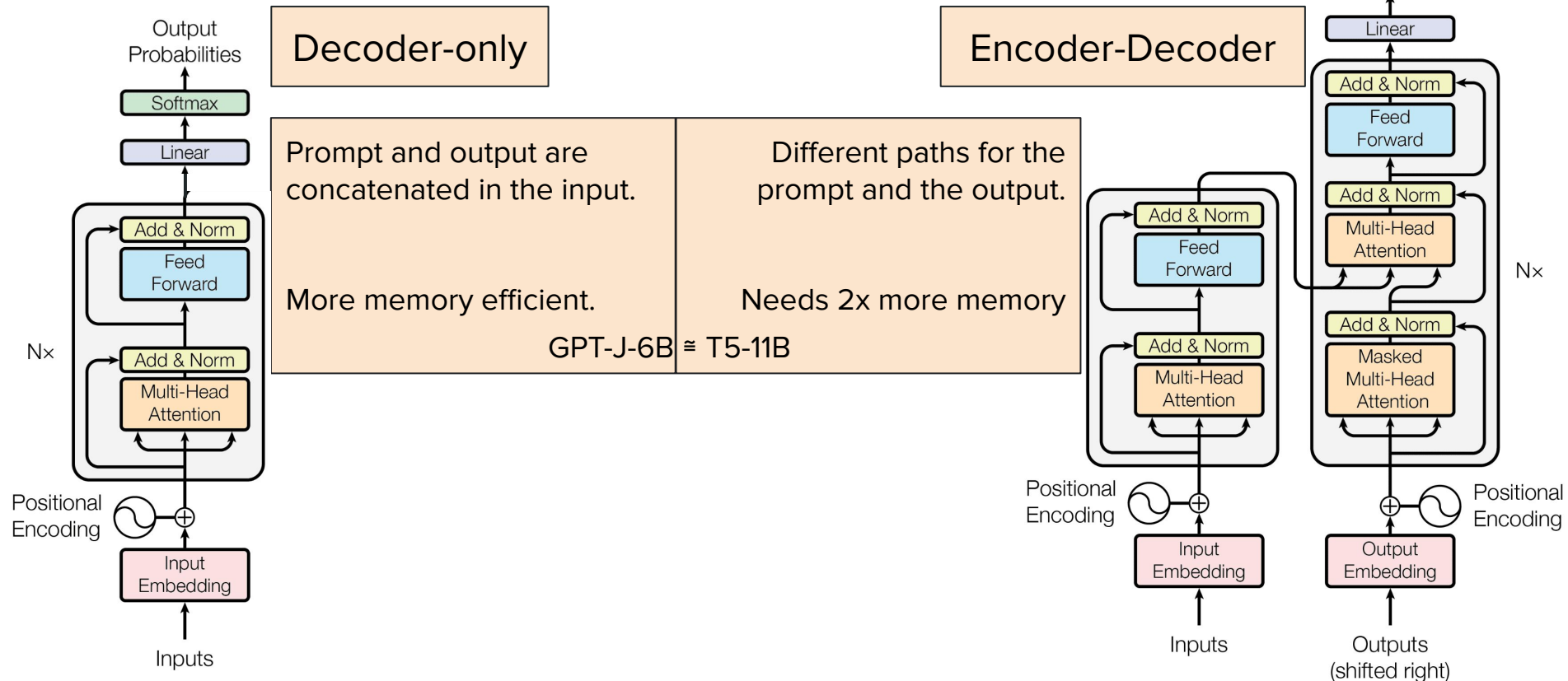
## Objectives:

- Causal LM
- Prefix LM
- Span Corruption

# Architectures - Vanilla Transformer



# Architectures - Vanilla Transformer



# Architectures - Transformer Variants

Narang et. al., showed that most transformer variants are comparable to vanilla transformers. We will not discuss them except Mixture of Experts (**MoEs**).

Mixture of Experts (**MoEs**) or Conditionally-Activated Transformers:

- Different examples activate different parts (“experts”) of the model
- Examples: Switch Transformer, Base-layer, DEMix Layers, GLaM, LaMDa

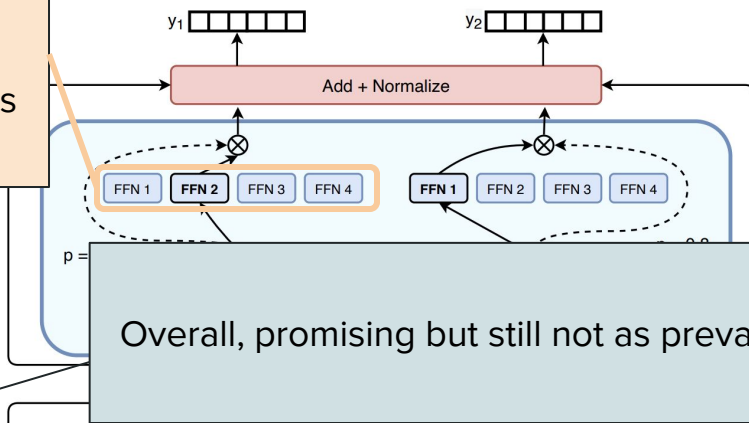
Narang et. al., 2021: Do Transformer Modifications Transfer Across Implementations and Applications?



# Architectures - Mixture of Experts (MoEs)

Experts specialize.  
Saves compute  
because only  
relevant expert is  
activated

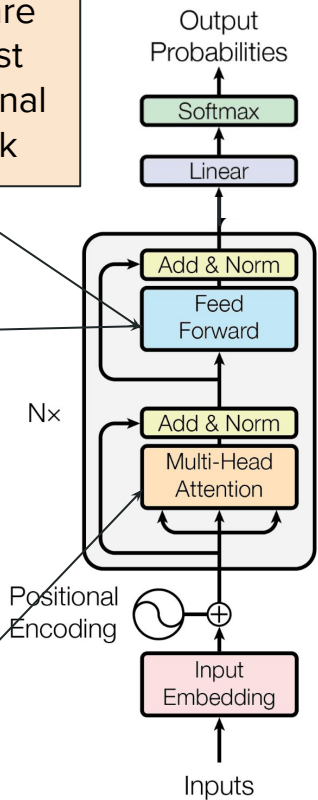
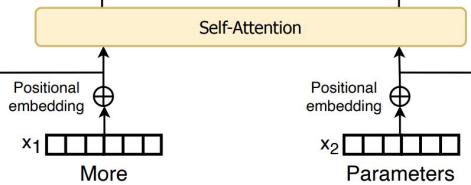
FF layers are  
the biggest  
computational  
bottleneck



Overall, promising but still not as prevalent as dense transformers

Router decides  
which expert to  
activate

The quadratic  
self-attention is  
negligible  
compared to FF  
in large LM



# Pretraining Objectives

## Causal LM (autoregressive LM)

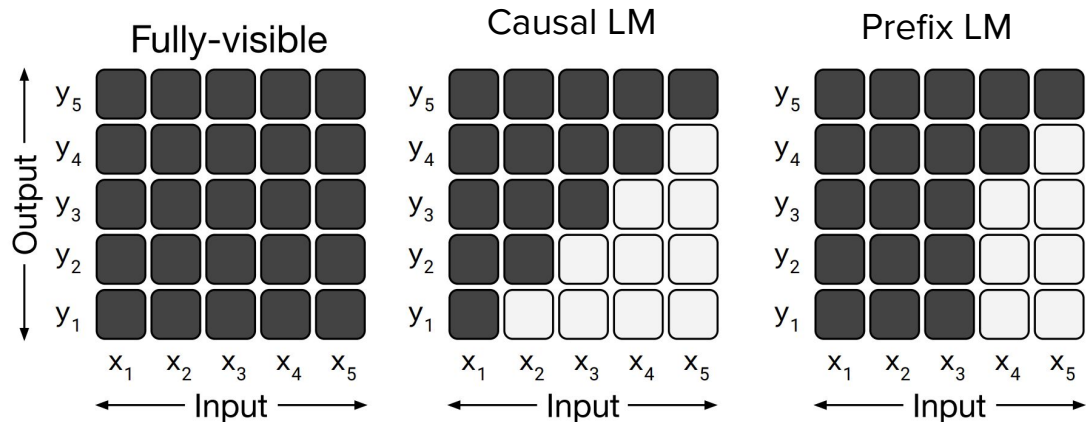
- Predict next word

## Prefix LM

- Predict next word given a fully-visible input

## Span Corruption

- Mask input spans then predict them in the output



Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

Inputs

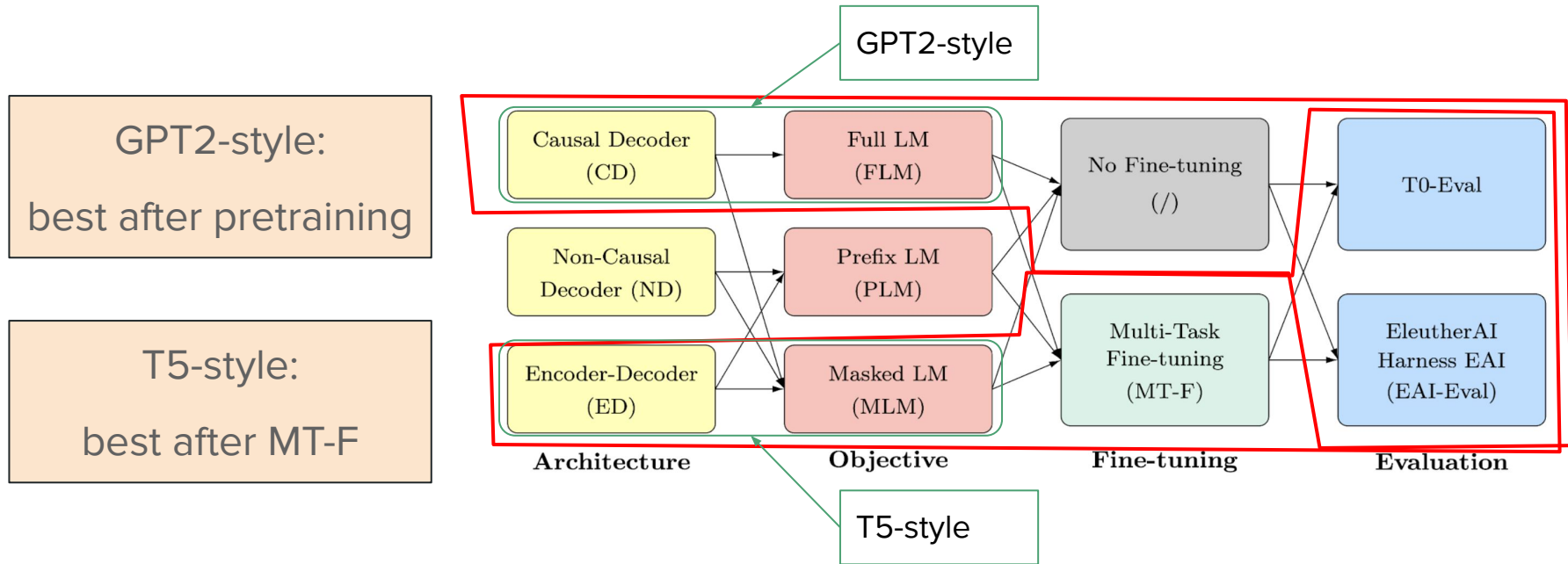
Thank you <X> me to your party <Y> week.

Targets

<X> for inviting <Y> last <Z>

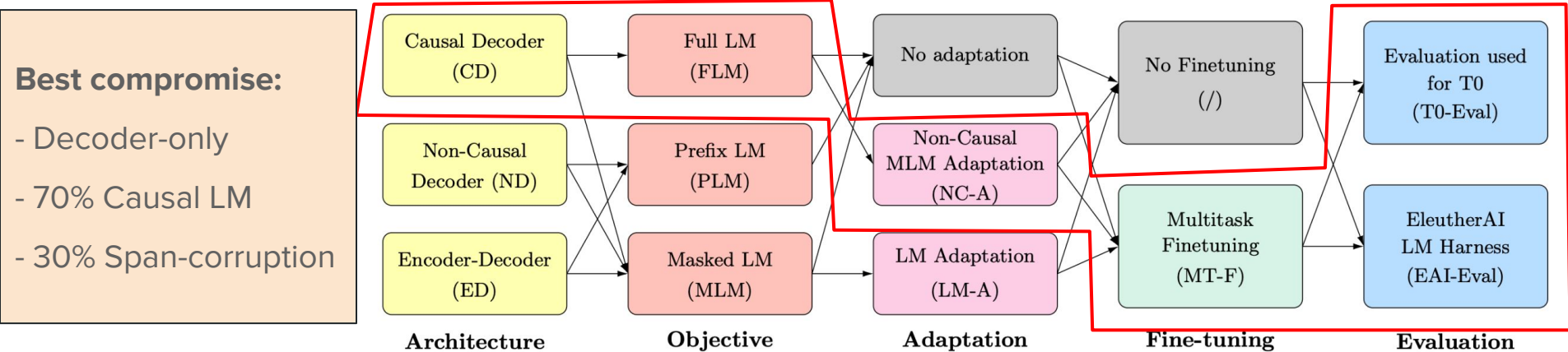
# Architecture and Pretraining Objective - Evaluation

Wang et. al., evaluated various combination for **zero-shot generalization**



# Architecture and Pretraining Objective - Evaluation

To get a single best model - add an LM adaptation phase on a different objective



# Tay et. al., 2022: Unifying Language Learning Paradigms

Argue that the objectives are complementary:

- Causal LM/Prefix LM: good for generation and ICL
- Span Corruption: good for multi-task finetuning

Recognize that Causal LM and Prefix LM are special cases of span corruption

- Both have a single very long masked span
- Define a single unified span-corruption function parameterized with
  - $\mu$ : average span length
  - $r$ : corruption rate
  - $n$ : number of corrupted spans

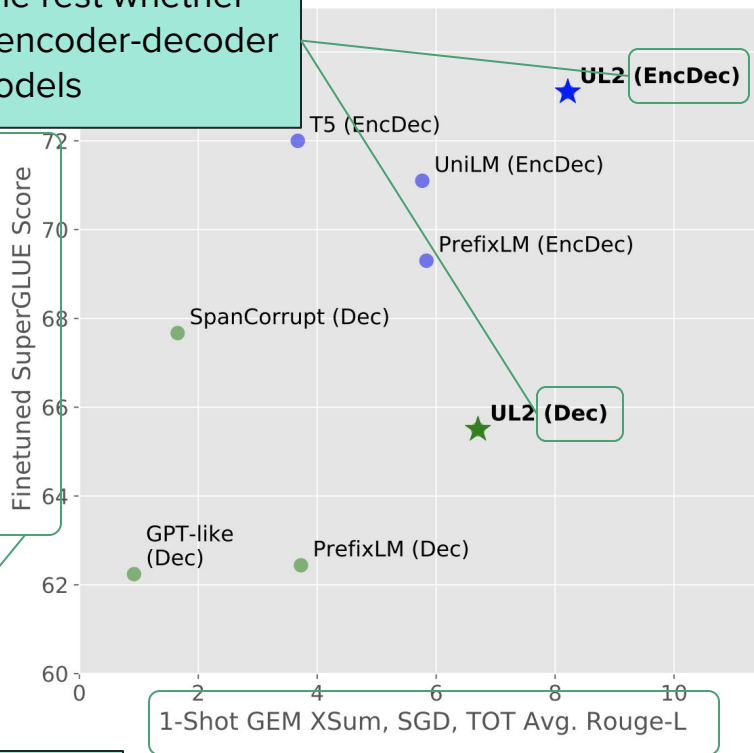
UL2: a mixture of 7 span-corruption objectives with different values of  $(\mu, r, n)$

# Tay et. al., 2022: Unifying Language Learning Paradigms

## Limitation:

- Missing the zero-shot evaluation with/without MT-F [as in Wang et. al., discussed earlier]
- Ignored causal LM
  - Claimed that Prefix LM is always better (not true)

UL2 outperforms the rest whether implemented with encoder-decoder or decoder-only models



T5-style are usually better

GPT2-style are usually better

# Architecture and Pretraining Objective

HTLM - BART-style model trained on HTML pages. HTML format enables fancy prompting methods

CM3 - a mix of autoregressive and span corruption to support generation and filling in space in one model

# Model Design - Caveats and Challenges

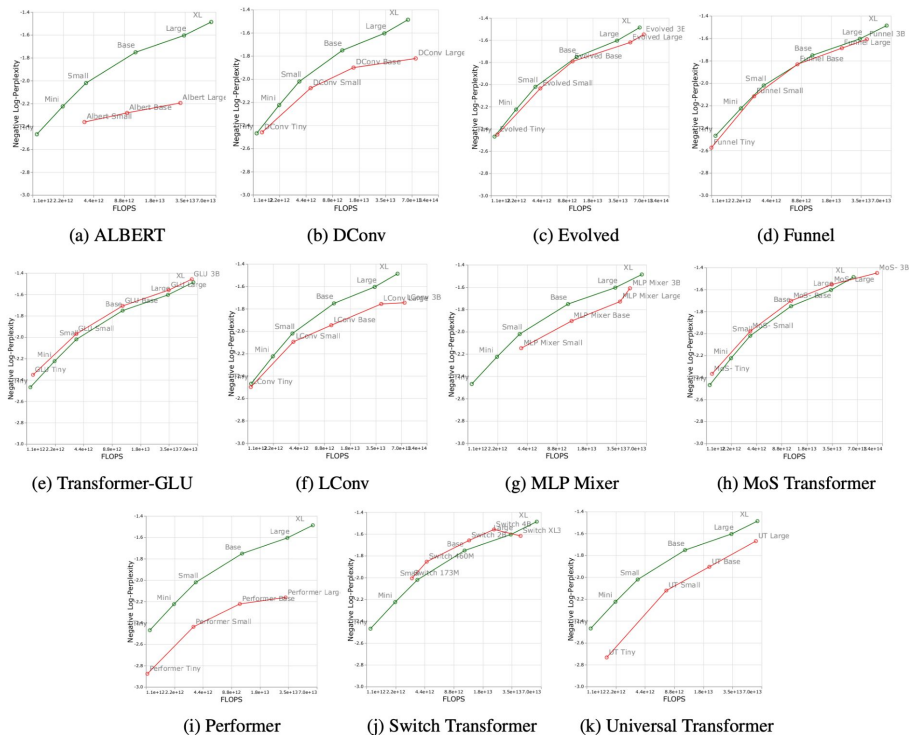
**Caveat 1** - modeling decisions and findings for small models don't necessarily scale.

Vanilla transformer vs. transformer alternative

Almost all models match or are worse than vanilla transformer.

Many modes get worse by scale.

⇒ Not enough to try new ideas on the small scale. Try the largest scale you can afford.





# Model Design - Caveats and Challenges

**Caveat 2** - New model designs should result in a **big** gain. If the gain is tiny, it is probably easier and cheaper to continue training the vanilla transformer a bit longer.

Cost should include

- Hyperparameter search
- Engineering cost to scale implementation
- Hardware throughout

⇒ Make sure the gains are significant enough they are worth implementing and scaling.

# Model Design - Caveats and Challenges

**Caveat 3** - How to you evaluate different modeling choices?

1. LM validation loss (scaling-laws work usually rely exclusively on this)
2. Downstream zero/few-shot in-context learning
3. Downstream few-shot using parameter-efficient finetuning
4. Downstream fully supervised full finetuning

Tay et. al., and Abnar et. al., showed that (1) and (4) are not always correlated.

Still not clear how strongly correlated (1) is with (2) and (3).

⇒ Include various downstream evaluation setups. LM validation loss isn't enough.

Tay et. al., 2022: Scale Efficiently: Insights From Pre-trained and Fine-tuned Transformers

Abnar et. al., 2021: Exploring the Limits of Large Scale Pre-training

# Overview

Previous sections: adapting an **existing** pretrained model to a new task.

This section: considerations for how to **build** these models to improve zero/few-shot performance

## Considerations

- Scaling
- Architecture and Objective
- **Dataset**
- Efficient pretraining and Engineering

# Dataset

Pretraining data makes a huge difference. Result from Scao et. al., 2022:

Model	Parameters	Pretraining tokens			
		Dataset	112B	250B	300B
<b>OpenAI</b> — Curie	6.7B			<u>49.28</u>	
<b>OpenAI</b> — Babbage	1.3B			<b>45.30</b>	
<b>EleutherAI</b> — GPT-Neo	1.3B	The Pile		42.94	
	13B	OSCAR		47.09	
<b>Ours</b> (BigScience)	1.3B	The Pile	<b>42.79</b>	43.12	<b>43.46</b>
	1.3B	C4	42.77		
	1.3B	OSCAR	41.72		

Scao et. al., 2022: What Language Model to Train if You Have One Million GPU Hours?

# Dataset

Very little work on dataset design for pretraining

- Mostly heuristic data filtering and deduplication
  - Gao 2021 - experimented with different levels of filtering
  - Katherine et. al., 2022 - experimented with different deduplication methods

With the results from Hoffmann et. al., dataset design becomes more important

- We need 10x more tokens than before

Gao 2021: An Empirical Exploration in Quality Filtering of Text Data

Katherine et. al., 2022: Deduplicating Training Data Makes Language Models Better

Hoffmann et. al., 2022: Training Compute-Optimal Large Language Models

# Overview

Previous sections: adapting an **existing** pretrained model to a new task.

This section: considerations for how to **build** these models to improve zero/few-shot performance

## Considerations

- Scaling
- Architecture and Objective
- Dataset
- **Efficient pretraining**

# Efficient pretraining

No direct impact on zero/few-shot downstream performance but more efficient pretraining translates to a better model using the same budget

Examples of efficient pretraining methods

- Curriculum learning: gradually increasing sequence length during training
- Staged training: gradually grow model size during training
- 8-bit Optimizers: saves memory at training time
- ...

# Engineering Considerations

No impact on zero/few-shot downstream performance but helps with pretraining

## Engineering

- Instabilities: fp16 or **bfloat16**
- Model parallelism: pipeline parallelism (PP), tensor parallelism (TP)
- Hardware: V100, A100, TPUs
- Hardware throughput
  - Small tweaks to ratio of depth/width lead to huge difference in throughput
- ...



# Summary & Open Questions

Scaling: scale the compute, find the optimal model size, and stop training at optimality

- Hoffmann et. al. made a tiny change to the LR schedule with huge impact to scaling. What else can improve the current recipe of training large language models?

Architecture and Objective

- No model works the best in all settings. Getting closer but more research is needed
- Make sure new model designs scale well

Dataset: It significantly impacts performance

- Explore new principled approaches to design larger & better datasets
- Understand the connection between pretraining data and downstream performance

Efficient pretraining: explore new methods

# Schedule

- 14:30–14:45 Part 1: Introduction [Sameer]
- 14:45–15:20 Part 2: Prompting & In-context learning [Sewon]
- 15:20–15:50 Part 3: Gradient-based LM task adaptation [Rob]
- 15:50–16:00 QnA for Part 1+2+3
  
- 16:00–16:30 Break
  
- 16:30–16:45 Part 4: Other methods of defining a task [Sameer]
- 16:45–17:05 Part 5: Evaluation benchmark [Arman]
- 17:05–17:25 Part 6: Meta-training [Arman]
- 17:25–17:45 Part 7: Pretraining considerations for zero/few-shot [Iz]
- 17:45–18:00 Conclusion/Future work + QnA [Iz]

# Conclusion/Future work

---

- Large language models made zero/few-shot a reality
- Zero/few-shot are interesting from the academic and practical perspectives.
- Prompting & In-context learning: learning a task without gradient updates
  - Future work: better in-context learning & better understanding why it works
- Parameter-efficient finetuning for task-adaptation
  - Future work: better finetuning methods & extend to tasks like generation
- Other methods of defining a task: instructions, prompts
- Evaluation benchmarks for zero/few-shot methods: principles and caveats
  - Future work: unified evaluation frameworks to allow direct comparisons
- Meta-training: training on supervised datasets
  - Future work: better understanding information learnt during meta-training, why some models work better than others
- Pretraining considerations: scale, architecture & objectives, and dataset
  - Future work: better datasets & understanding connection to performance, efficient pretraining methods, and models that work across all settings

# Questions?

Slides available at:

<https://github.com/allenai/acl2022-zerofewshot-tutorial>

Part 1: Introduction [Sameer]

Part 2: Prompting & In-context learning [Sewon]

Part 3: Gradient-based LM task adaptation [Rob]

Part 4: Other methods of defining a task [Sameer]

Part 5: Evaluation benchmark [Arman]

Part 6: Meta-training [Arman]

Part 7: Pretraining considerations for zero/few-shot [Iz]

Part 8: Future work

# Prompt-based finetuning

How to address multi-token verbalizers for masked language models:

1. Don't use them.
2. **Hinge loss** (training) + Autoregressive decoding (inference)

$x = \text{Awful pizza! It was } [\text{MASK}]_1 [\text{MASK}]_2 .$

$p_+ = p([\text{MASK}]_1 = \text{terri} \mid x) * p([\text{MASK}]_2 = \text{\#ble} \mid x)$

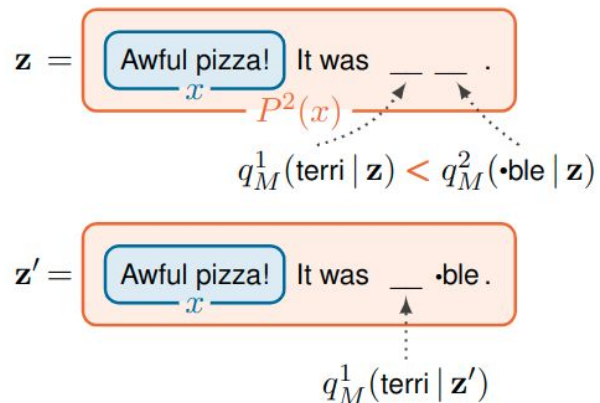
$p_- = p([\text{MASK}]_1 = \text{great} \mid x)$

$L = \max(0, 1 - \log p_+ + \log p_-)$

# Prompt-based finetuning

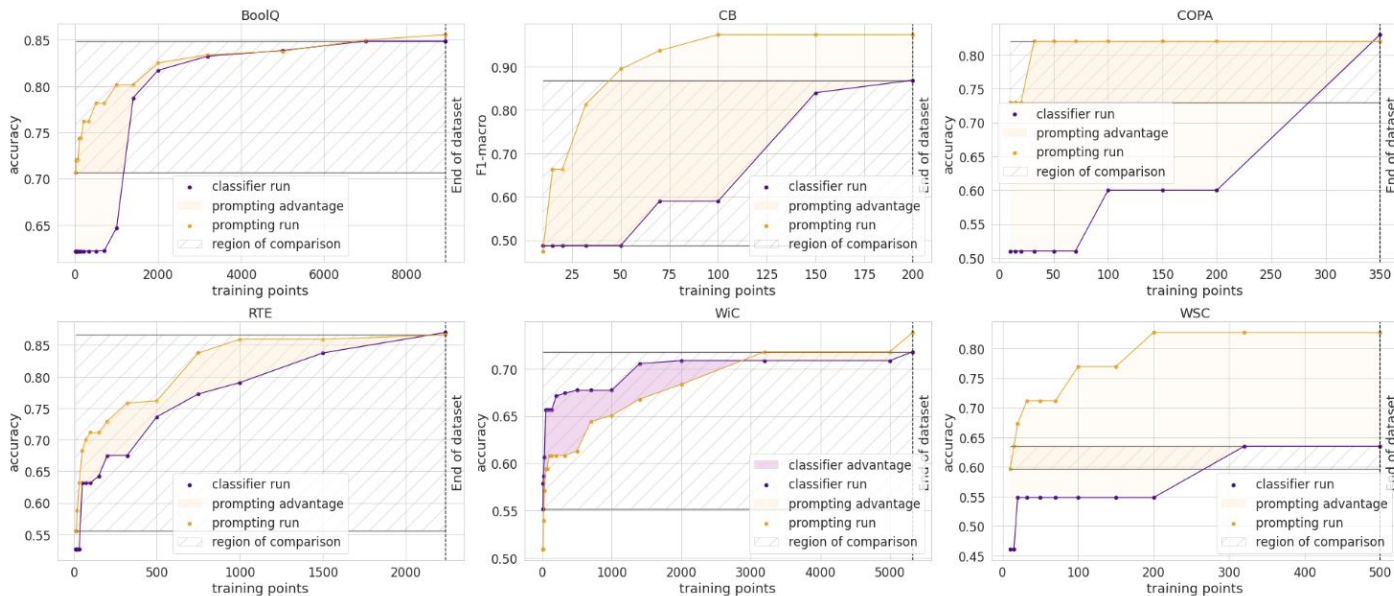
How to address multi-token verbalizers for masked language models:

1. Don't use them.
2. Hinge loss (training) + **Autoregressive decoding** (inference)



# Prompt-based finetuning vs. traditional finetuning

Prompt-based finetuning has higher few-shot accuracy than traditional finetuning.



ALBERT<sub>XXLARGE-V2</sub>



# Prompt-based finetuning vs. traditional finetuning

Prompt-based finetuning has higher few-shot accuracy than traditional finetuning.

	<b>SST-2</b> (acc)	<b>SST-5</b> (acc)	<b>MR</b> (acc)	<b>CR</b> (acc)	<b>MPQA</b> (acc)	<b>Subj</b> (acc)	<b>TREC</b> (acc)	<b>CoLA</b> (Matt.)
Fine-tuning	81.4 (3.8)	43.9 (2.0)	76.9 (5.9)	75.8 (3.2)	72.0 (3.8)	90.8 (1.8)	88.8 (2.1)	<b>33.9</b> (14.3)
Prompt-based FT (man)	<u>92.7 (0.9)</u>	<u>47.4 (2.5)</u>	<u>87.0 (1.2)</u>	<u>90.3 (1.0)</u>	<u>84.7 (2.2)</u>	<u>91.2 (1.1)</u>	84.8 (5.1)	9.3 (7.3)

	<b>MNLI</b> (acc)	<b>MNLI-mm</b> (acc)	<b>SNLI</b> (acc)	<b>QNLI</b> (acc)	<b>RTE</b> (acc)	<b>MRPC</b> (F1)	<b>QQP</b> (F1)	<b>STS-B</b> (Pear.)
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-based FT (man)	<u>68.3 (2.3)</u>	<u>70.5 (1.9)</u>	<u>77.2 (3.7)</u>	<u>64.5 (4.2)</u>	<u>69.1 (3.6)</u>	74.5 (5.3)	<u>65.5 (5.3)</u>	<u>71.0 (7.0)</u>

# PET - Results

Knowledge distillation improves accuracy.

Method	Yelp	AG's	Yahoo	MNLI
min	39.6	82.1	50.2	36.4
max	52.4	85.0	63.6	40.2
PET (no distillation)	51.7	87.0	62.8	40.6
PET uniform	52.7	87.3	<b>63.8</b>	<b>42.0</b>
PET weighted	<b>52.9</b>	<b>87.5</b>	<b>63.8</b>	41.8

RoBERTa<sub>LARG</sub>

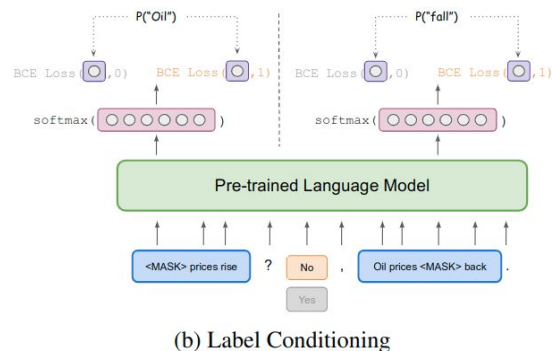
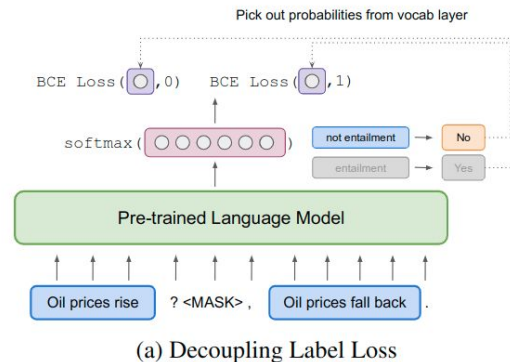
E

Table 4: Minimum (min) and maximum (max) accuracy of models based on individual PVPs as well as PET with and without knowledge distillation ( $|\mathcal{T}| = 10$ ).

# ADAPET

Modifications to previous setup:

- **Decoupling Label Loss:** Includes all LM probabilities instead of just verbalizers.
- **Label Conditioning:** Use probability of context given labels (similar to noisy channel).
- **No Unlabeled Data**



# Sensitivity to hyperparameters

ADAPET sensitive to # of gradient updates and fraction of masked words.

Performance drops when hyperparameters are chosen using model selection criteria, e.g., cross validation (CV) and minimum description length (MDL).

	<b>BoolQ</b> Acc	<b>CB</b> Acc/F1	<b>COPA</b> Acc	<b>RTE</b> Acc	<b>WiC</b> Acc	<b>WSC</b> Acc	<b>MultiRC</b> EM/F1	<b>ReCoRD</b> EM/F1	<b>Avg</b>
<b>Worst</b>	75.0 <sub>4.8</sub>	79.5 <sub>2.3</sub> /67.3 <sub>7.8</sub>	76.8 <sub>2.2</sub>	63.2 <sub>4.0</sub>	49.0 <sub>1.3</sub>	77.2 <sub>1.8</sub>	38.5 <sub>7.4</sub> /80.0 <sub>2.9</sub>	76.2 <sub>1.8</sub> /86.5 <sub>1.2</sub>	69.4 <sub>1.5</sub>
<b>Mean</b>	79.0 <sub>1.5</sub>	85.9 <sub>2.3</sub> /74.5 <sub>11.0</sub>	81.1 <sub>2.9</sub>	70.8 <sub>2.5</sub>	51.5 <sub>1.8</sub>	82.5 <sub>2.7</sub>	44.2 <sub>6.6</sub> /82.3 <sub>2.7</sub>	78.3 <sub>1.3</sub> /87.8 <sub>0.8</sub>	73.9 <sub>1.2</sub>
<b>MDL</b>	76.5 <sub>5.8</sub>	85.7 <sub>5.6</sub> /74.8 <sub>13.4</sub>	82.0 <sub>2.9</sub>	70.4 <sub>8.5</sub>	52.2 <sub>3.0</sub>	82.0 <sub>3.1</sub>	39.7 <sub>8.1</sub> /80.6 <sub>3.2</sub>	78.9 <sub>0.7</sub> /88.2 <sub>0.4</sub>	73.4 <sub>2.8</sub>
<b>CV</b>	78.9 <sub>2.4</sub>	83.9 <sub>5.3</sub> /69.2 <sub>10.3</sub>	80.5 <sub>3.3</sub>	68.7 <sub>7.0</sub>	51.1 <sub>1.6</sub>	83.1 <sub>2.6</sub>	41.9 <sub>7.2</sub> /81.4 <sub>3.1</sub>	78.7 <sub>1.6</sub> /88.1 <sub>1.0</sub>	73.0 <sub>2.1</sub>
<b>Best</b>	80.9 <sub>1.0</sub>	89.8 <sub>3.1</sub> /79.8 <sub>13.4</sub>	84.8 <sub>4.5</sub>	76.7 <sub>1.8</sub>	54.1 <sub>2.3</sub>	86.6 <sub>1.8</sub>	46.8 <sub>6.9</sub> /83.4 <sub>2.9</sub>	80.4 <sub>1.1</sub> /89.2 <sub>0.7</sub>	77.2 <sub>0.9</sub>
<b>ADAPET</b> [12]	80.3	89.3 / 86.8	89.0	76.5	54.4	81.7	39.2 / 80.1	85.4 / 92.1	77.3
<b>iPET</b> [9]	80.6	92.9 / 92.4	95.0	74.0	52.2	80.1	33.0 / 74.0	86.0 / 86.5	76.8
<b>PET</b> [9]	79.4	85.1 / 59.4	95.0	69.8	52.4	80.1	37.9 / 77.3	86.0 / 86.5	74.1
<b>GPT-3</b> [2]	77.5	82.1 / 57.2	92.0	72.9	55.3	75.0	32.5 / 74.8	89.0 / 90.1	73.2

ALBERT  
XXLARGE-V2

# Sensitivity to prompt

Webson and Pavlick (2021) additionally study the effect of irrelevant and misleading prompts on accuracy.

Category	Examples
instructive	{premise} Are we justified in saying that "{hypo}"? Suppose {premise} Can we infer that "{hypo}"?
misleading-moderate	{premise} Can that be paraphrased as: "{hypo}"? {premise} Are there lots of similar words in "{hypo}"?
misleading-extreme	{premise} is the sentiment positive? {hypo} {premise} is this a sports news? {hypo}
irrelevant	{premise} If bonito flakes boil more than a few seconds the stock becomes too strong. "{hypo}"?
null	{premise} {hypothesis} {hypothesis} {premise}

Table 1: Example templates for NLI.

# Sensitivity to prompt

Webson and Pavlick (2021) additionally study the effect of irrelevant and misleading prompts on accuracy, and find there can be “no practical difference”.

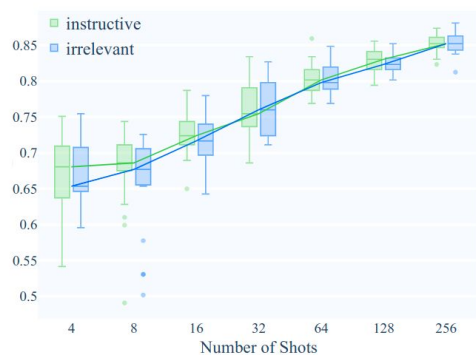


Figure 2: T0 (3B) on RTE. There is no practical difference between the performance of the models trained with instructive templates vs. those trained with irrelevant templates at any number of shots.

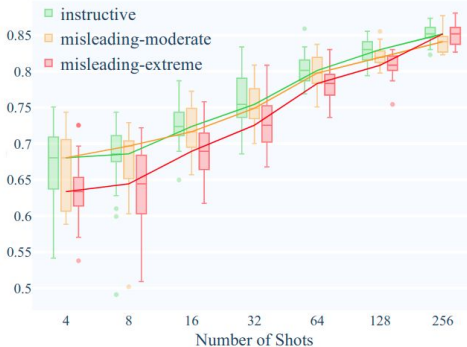


Figure 3: T0 (3B) on RTE. There is no practical difference between models trained with instructive and misleading-moderate templates at any number of shots. But models trained with misleading-far templates are statistically significantly worse from 8 to 128 shots.

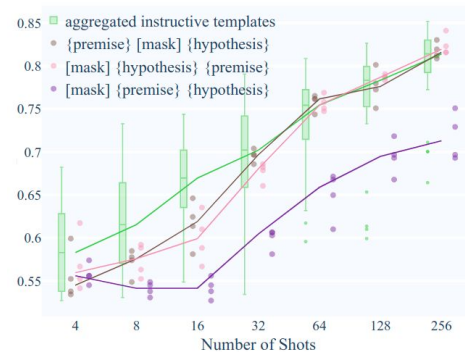


Figure 4: ALBERT on RTE. After 32 shots, models trained with 2 null templates learn just as fast as the instructive templates, but models trained with other null templates (e.g., purple) are much worse.

# Sensitivity to prompt

Webson and Pavlick (2021) additionally study the effect of irrelevant and misleading prompts on accuracy, and find there can be “no practical difference”.

	size	#shots	inst. > mis-moderate	inst. > mis-extreme	inst. > irrelevant	inst. > null
ALBERT	235M	4 - 256	✓			✓
T5 LMA	770M	4 - 256				
T5 LMA	3B	4 - 256	✓			✓
T0	3B	4 - 256		✓		✓
T5 LMA	11B	16	✓	✓		✓
T0	11B	16		✓		✓
T0++	11B	16	✓			✓
GPT-3	175B	16				✓

Table 2: Checkmarks indicate where two categories of templates lead to statistically significantly different performance, as measured by an independent two-sample  $t$ -test and a Wilcoxon rank-sum test; both tests always agree in this table.

# Sensitivity to prompt

Schick and Schütze (2021) compare the per-prompt accuracies of 4 different types of prompt.

NULL

[MASK]  $x_1 x_2$

PUNC

[MASK] :  $x_1 x_2$

PROMPT

$x_1 x_2$  Topic: [MASK]

Q&A

$x_1 x_2$  Question: What is the topic of this article?  
Answer: [MASK]



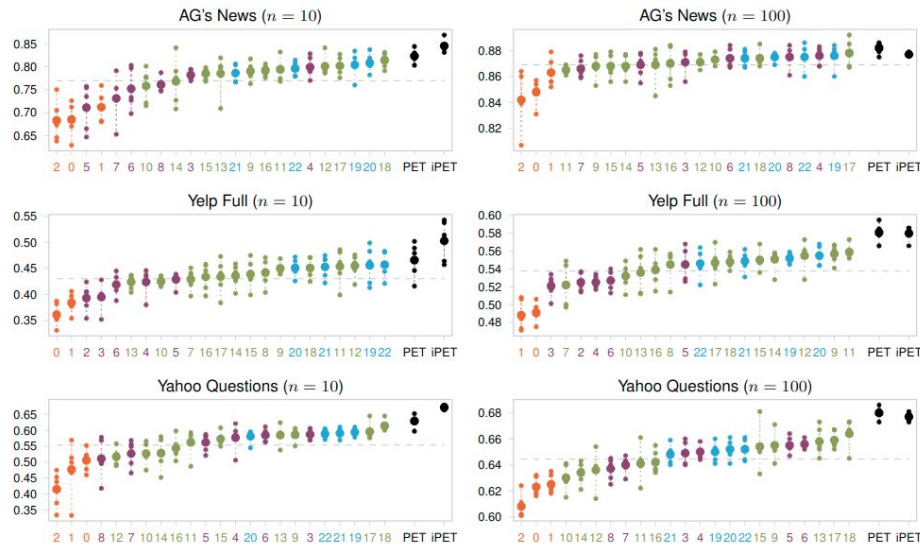
# Sensitivity to prompt

Schick and Schütze (2021) compare the per-prompt accuracies of 4 different types of prompt, and find that instructive prompts often rank higher than other prompts.

(NULL, PUNC, PROMPT, Q&A)

Small points = individual training sets.  
Big points = average over training sets.

RoBERTa<sub>LARG</sub>  
E



# Sensitivity to prompt / hyperparameters

The RAFT benchmark lacks publicly available data for tuning.

PET achieves near-human performance on 7 of 11 tasks.

Method	ADE	B77	NIS	OSE	Over	SOT	SRI	TAI	ToS	TEH	TC	Avg
GPT-2	60.0	12.1	56.1	24.5	49.8	38.0	49.2	61.2	49.8	31.1	72.3	45.8
GPT-Neo	45.2	14.9	40.8	34.3	68.1	40.6	49.3	60.5	56.5	55.4	63.6	48.1
AdaBoost	54.3	02.3	62.6	47.5	83.8	45.5	50.6	55.6	56.0	44.3	62.5	51.4
snlt	60.3	24.8	58.5	30.2	83.1	33.6	49.2	62.6	54.0	44.9	79.1	52.8
GPT-3	68.6	29.9	67.9	43.1	<b>93.7</b>	76.9	<b>51.6</b>	<b>65.6</b>	57.4	52.6	82.1	62.7
SetFit	72.6	53.8	<b>87.2</b>	52.1	90.7	68.2	49.3	62.8	<b>62.0</b>	<b>53.2</b>	<b>83.7</b>	66.9
PET	<b>82.2</b>	<b>59.3</b>	85.7	<b>64.6</b>	90.8	<b>81.6</b>	49.3	63.8	57.6	48.3	82.4	<b>69.6</b>
Human	<u>83.0</u>	<u>60.7</u>	85.7	<u>64.6</u>	91.7	<u>90.8</u>	46.8	60.9	<u>62.7</u>	<u>72.2</u>	<u>89.7</u>	<u>73.5</u>

ALBERT<sub>XXLARGE-V2</sub>

50 Training data  
points

Table 1: Performance of various baselines and PET on the RAFT benchmark (Alex et al., 2021); shown numbers are macro F1 scores multiplied by 100. Best model performance is shown in bold, best overall performance (including human annotators) is underlined. The final column shows average performance across all 11 tasks.

# Sensitivity to prompt

To summarize:

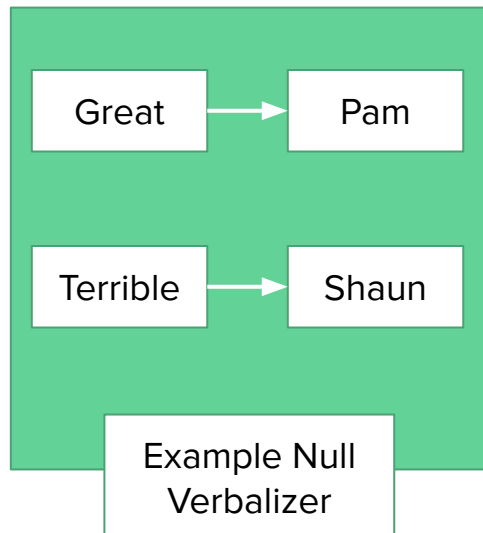
Instructive prompts are shown to have higher accuracy in most cases.

However, the effect size is often small relative to variation in accuracy due to random seed, etc.

There is often no significant difference between instructive prompts and null or misleading prompts.

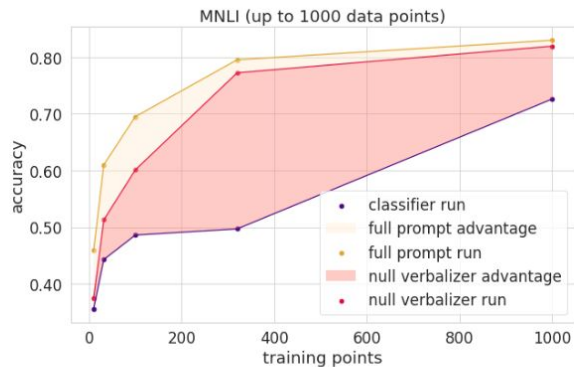
# Sensitivity to verbalizer

Le Scao and Rush (2021) study the average prompting advantage of “null verbalizers” that replace the verbalizers (a.k.a. label tokens) with randomly chosen first names.



# Sensitivity to verbalizer

They find “for small data tasks...the null verbalizer removes much of the benefits of prompting”.



RoBERTa<sub>LARG</sub>

E

# Sensitivity to verbalizer

Webson and Pavlick (2021) similarly find that flipped and nonsense labels are detrimental to performance in few-shot settings.

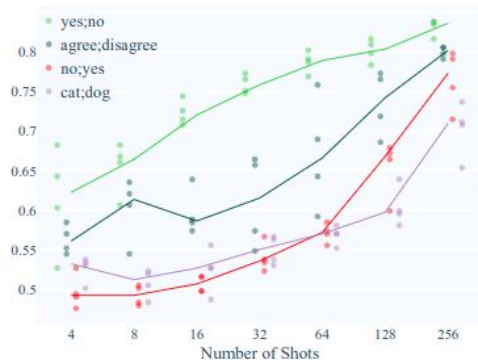


Figure 7: The best-performing instructive template for ALBERT on RTE, {prem} Are we justified in saying that "{hypo}"? with select LM targets from each category.

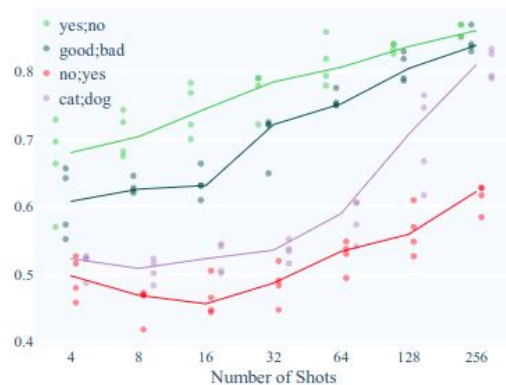
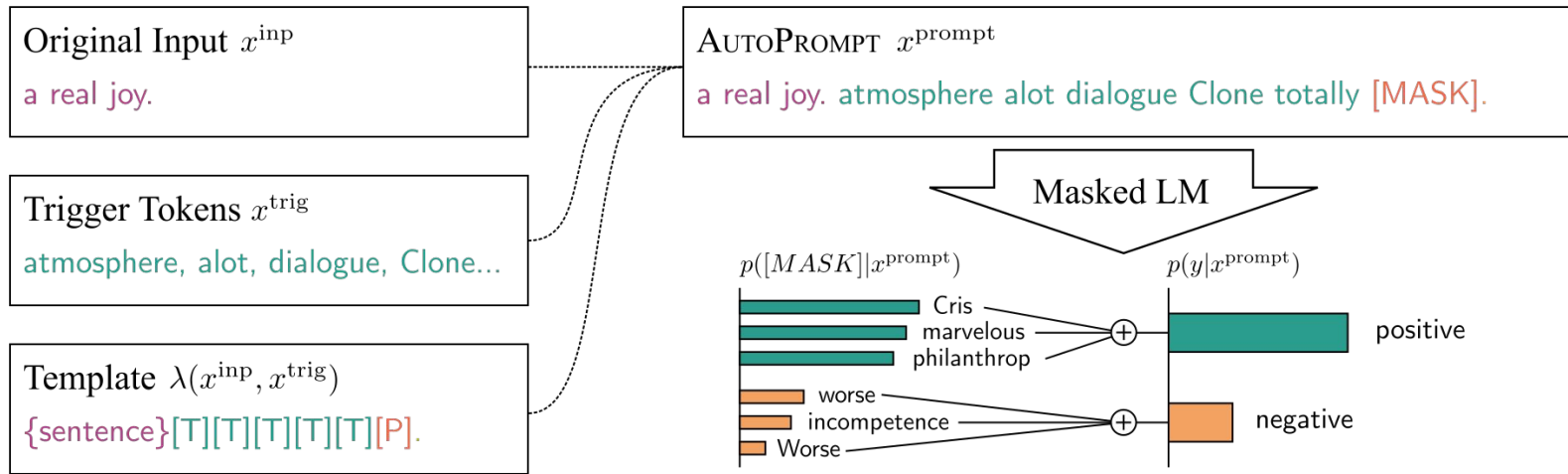


Figure 9: The best-performing instructive template for T0 (3B) on RTE, {prem} Based on the previous passage, is it true that "{hypo}"? with select LM targets from each category.

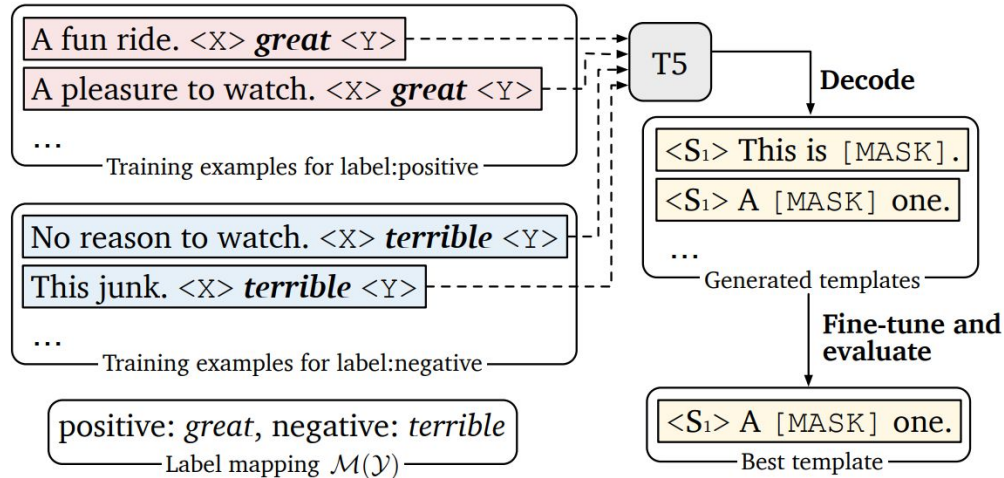
# AutoPrompt

AutoPrompt learns the tokens in the pattern using a gradient guided search.



# LM-BFF

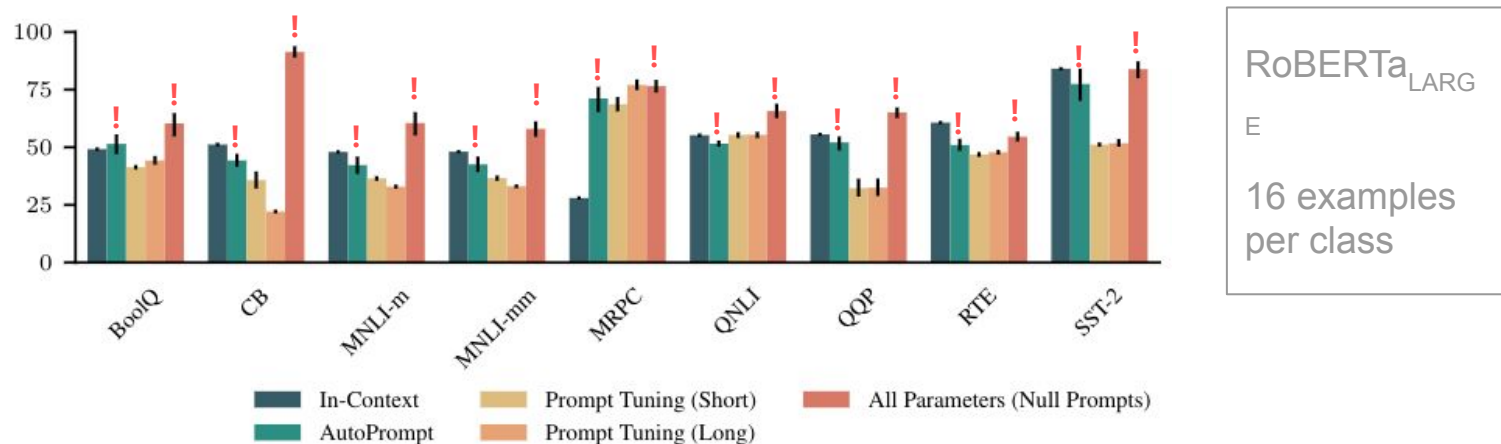
LM-BFF uses T5's span filling capabilities to write prompts.





# AutoPrompt - Results

Comparison of AutoPrompt and prompt-based finetuning (w/ null prompts) in few-shot settings.



# LM-BFF - Results

Comparison of automated and manually written prompts in a *prompt-based finetuning* setting.

	SST-2 (acc)	SST-5 (acc)	MR (acc)	CR (acc)	MPQA (acc)	Subj (acc)	TREC (acc)	CoLA (Matt.)
Majority <sup>†</sup>	50.9	23.1	50.0	50.0	50.0	50.0	18.8	0.0
Prompt-based zero-shot <sup>‡</sup>	83.6	35.0	80.8	79.5	67.6	51.4	32.0	2.0
“GPT-3” in-context learning	84.8 (1.3)	30.6 (0.9)	80.5 (1.7)	87.4 (0.8)	63.8 (2.1)	53.6 (1.0)	26.2 (2.4)	-1.5 (2.4)
Fine-tuning	81.4 (3.8)	43.9 (2.0)	76.9 (5.9)	75.8 (3.2)	72.0 (3.8)	90.8 (1.8)	88.8 (2.1)	<b>33.9</b> (14.3)
→ Prompt-based FT (man) + demonstrations	92.7 (0.9)	47.4 (2.5)	87.0 (1.2)	90.3 (1.0)	84.7 (2.2)	91.2 (1.1)	84.8 (5.1)	9.3 (7.3)
→ Prompt-based FT (auto) + demonstrations	92.3 (1.0)	49.2 (1.6)	85.5 (2.8)	89.0 (1.4)	85.8 (1.9)	91.2 (1.1)	88.2 (2.0)	14.0 (14.1)
	<b>93.0</b> (0.6)	49.5 (1.7)	<b>87.7</b> (1.4)	<b>91.0</b> (0.9)	86.5 (2.6)	91.4 (1.8)	<b>89.4</b> (1.7)	21.8 (15.9)
Fine-tuning (full) <sup>†</sup>	95.0	58.7	90.8	89.4	87.8	97.0	97.4	62.6

RoBERTa<sub>LARG</sub>

E

16 examples  
per class

# Summary of input-level modifications

	Pattern	LM	Few-Shot Experiments (Prompt Only)	Few-Shot Experiments (Full Finetuning)
AutoPrompt	Discrete	*BERT	✓	✗
LM-BFF	Discrete	*BERT	✗	✓
Prefix Tuning	Continuous (Deep)	GPT-2 / BART	✓	✗
WARP	Continuous	*BERT	✓	✗
P-Tuning	Continuous	GPT-2 / *BERT	✗	✓
OptiPrompt	Continuous	*BERT	✗	✗
Soft Prompts	Continuous (Deep)	*BERT / BART	✗	✗
Prompt Tuning	Continuous	T5	✗	✗

# Continuous Prompt Search

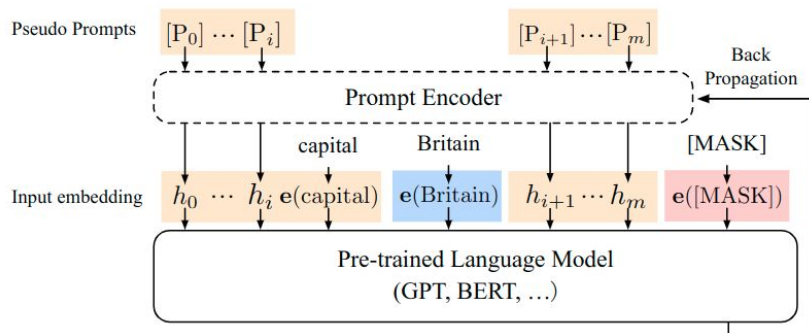
Many proposed approaches:

- **Prefix Tuning\*** (Li and Liang, 2021. “Prefix-Tuning: Optimizing Continuous Prompts for Generation”)
- **WARP** (Hambardzumyan et al., 2021. “WARP: Word-level Adversarial ReProgramming”)
- **P-Tuning** (Liu et al., 2021. “GPT Understands, Too”)
- **OptiPrompt** (Zhong et al., 2021. “Factual Probing Is [MASK]: Learning vs. Learning to Recall”)
- **Soft Prompts\*** (Qin and Eisner, 2021. “Learning How to Ask: Querying LMs with Mixtures of Soft Prompts”)
- **Prompt Tuning** (Lester et al., 2021. “The Power of Scale for Parameter-Efficient Prompt Tuning”)

\* Tunes contextualized embeddings as well.

# Continuous Prompt Search

P-Tuning uses an LSTM layer to encode the “dummy” token embeddings before sending to the language model.



# Prompt Tuning - Results

		<b>Model</b>	<b>CB</b>	<b>RTE</b>
			F <sub>1</sub> / Acc.	Acc.
dev		GPT-3 Small	26.1 / 42.9	52.3
		GPT-3 Med	40.4 / 58.9	48.4
		GPT-3	57.2 / 82.1	72.9
		PET (ALBERT)	59.4 / 85.1	69.8
		iPET (ALBERT)	92.4 / 92.9	74.0
		WARP <sub>init</sub> (ALBERT)	84.0 / 87.5	71.8
test		GPT-3	52.0 / 75.6	69.0
		PET (ALBERT)	60.2 / 87.2	67.2
		iPET (ALBERT)	79.9 / 88.8	70.8
		WARP <sub>init</sub> (ALBERT)	70.2 / 82.4	69.1

# Continuous Prompt Search

Using a noisy channel approach improves accuracy of prompt tuning for GPT-2.

Data	Direct			Channel
	Head	Trans	Prompt	Prompt
SST-2	80.2/68.6	77.3/57.5	72.6/50.9	<b>85.8/81.3</b>
SST-5	34.9/30.0	33.0/25.5	30.9/19.1	<b>36.3/27.9</b>
MR	73.7/ <b>56.4</b>	71.3/51.6	67.4/50.1	<b>81.7/78.0</b>
CR	67.6/50.0	63.9/50.0	65.7/50.0	<b>79.6/76.4</b>
Amazon	34.5/28.8	32.1/18.2	31.2/20.0	<b>43.4/39.2</b>
Yelp	40.6/32.8	38.9/31.5	31.9/20.6	<b>43.9/37.2</b>
TREC	<b>54.1/42.4</b>	48.0/31.0	35.9/13.0	37.1/20.8
AGNews	<b>74.1/61.2</b>	66.9/47.0	61.9/25.2	73.4/ <b>63.9</b>
Yahoo	39.1/31.4	33.8/23.0	27.4/15.7	<b>54.0/46.7</b>
DBPedia	49.3/37.5	42.4/28.6	41.8/9.9	<b>67.7/52.9</b>
Subj	<b>86.3/79.1</b>	86.0/71.6	65.5/49.9	75.5/58.8
Avg.	57.7/47.1	54.0/39.6	48.4/29.5	<b>61.7/53.0</b>

# Summary

Methods that learn discrete or continuous prompts:

- Allow multi-task batching
- Introduce minimal additional parameters

However, most works find that prompt tuning does not perform as well as prompt-based finetuning.