Large Language Models (LLMs)

Fine-Tuning

Learning goals

- comprehend the different subtleties in the space of fine-tuning and prompting

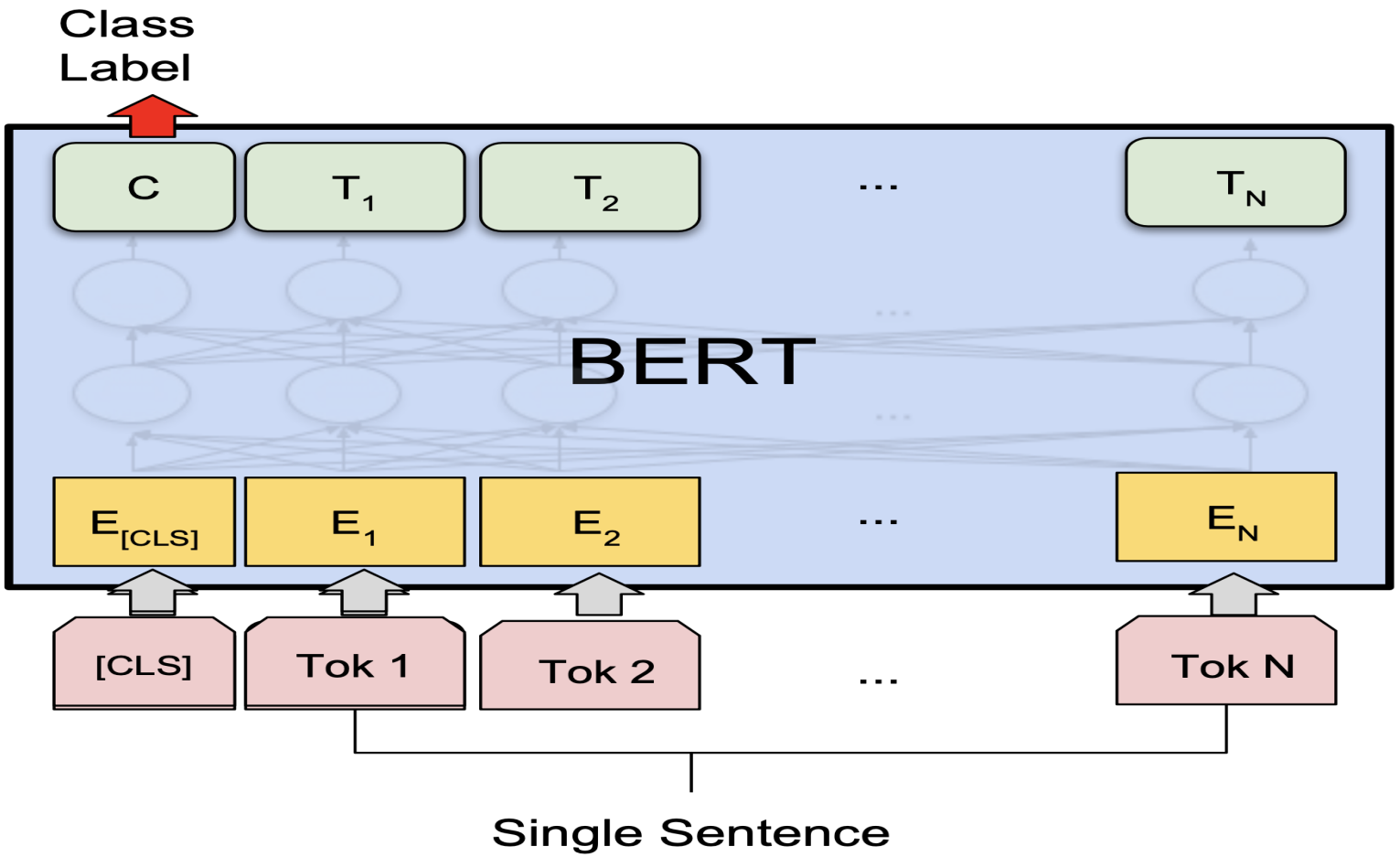
RECAP

- language modeling objectives
  - token prediction
  - no explicit understanding of tasks
- this is true for encoder and decoder models
- So we need to do additional work if we want to use language models for solving tasks!

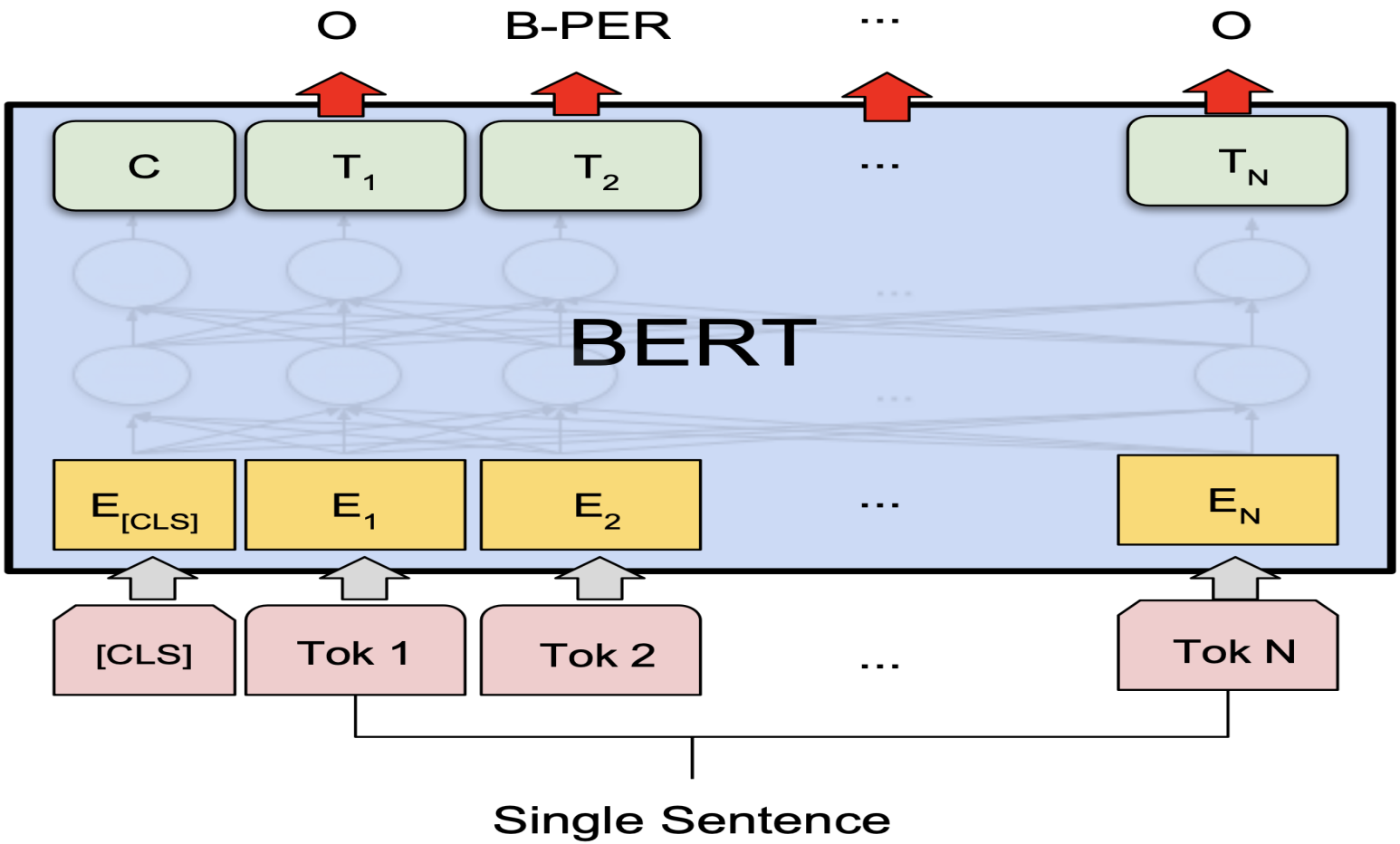
HOW TO SOLVE A TASK WITH LANGUAGE MODELS? (1)

- "old-style" single-task fine-tuning
  - supervised training of the pretrained model on a task-specific training set of size k (where k is not small, e.g., $k = 100$)

BERT FINETUNING EXAMPLE 1



BERT FINETUNING EXAMPLE 2



HOW TO SOLVE A TASK WITH LANGUAGE MODELS? (1)

- "old-style" single-task fine-tuning
  - supervised training of the pretrained model on a task-specific training set of size k (where k is not small, e.g., $k = 100$)
  - the output can be an arbitrary category (e.g., "0", "1" for sentiment analysis)
  - alternatively, the output can be a meaningful "verbalizer" (e.g., "negative", "positive" for sentiment analysis) ▸ Schick et al., 2020
  - this was the typical way encoder models like BERT are used
  - still very much relevant if you need to deploy a small efficient model for a focused task

VERBALIZER


INPUT
I really enjoyed this movie.
PATTERNS
I really enjoyed this movie. It was [MASK].
I really enjoyed this movie. Question: Is this a positive movie review? Answer: [MASK].
([MASK]) I really enjoyed this movie.
VERBALIZERS
+ = good − = bad
+ = Yes − = No
+ = positive − = negative

HOW TO SOLVE A TASK WITH LANGUAGE MODELS? (2)

- few-shot prompting
  - provide, in-context, k (where k is small, e.g., $k = 5$) examples of what the model is supposed to do
  - the model will then often complete the task just based on analogy to these few shots
  - this is the most typical way of using current autoregressive models for tasks
  - no change to the parameters of the model, i.e., no training

FEW-SHOT PROMPTING EXAMPLE

```
1  Translate English to French:
2  sea otter => loutre de mer
3  peppermint => menthe poivrée
4  plush girafe => girafe peluche
5  cheese =>
```

(from gpt3 paper – “Translate English to French” should appear three times to make it a 3-shot example that has 3 identical shots as required by the definition of few-shot prompting)

HOW TO SOLVE A TASK WITH LANGUAGE MODELS? (3)

- multi-task finetuning
  - finetuning on a large training set of *many* tasks
  - the more tasks the better
  - method 1: consistent task format, e.g., each task is reformulated into a question-answering format
  - method 2: more open task format
  - ideally: task format is consistent with language model objective (similar to verbalizer)
  - Examples: T5 and FLAN, see below

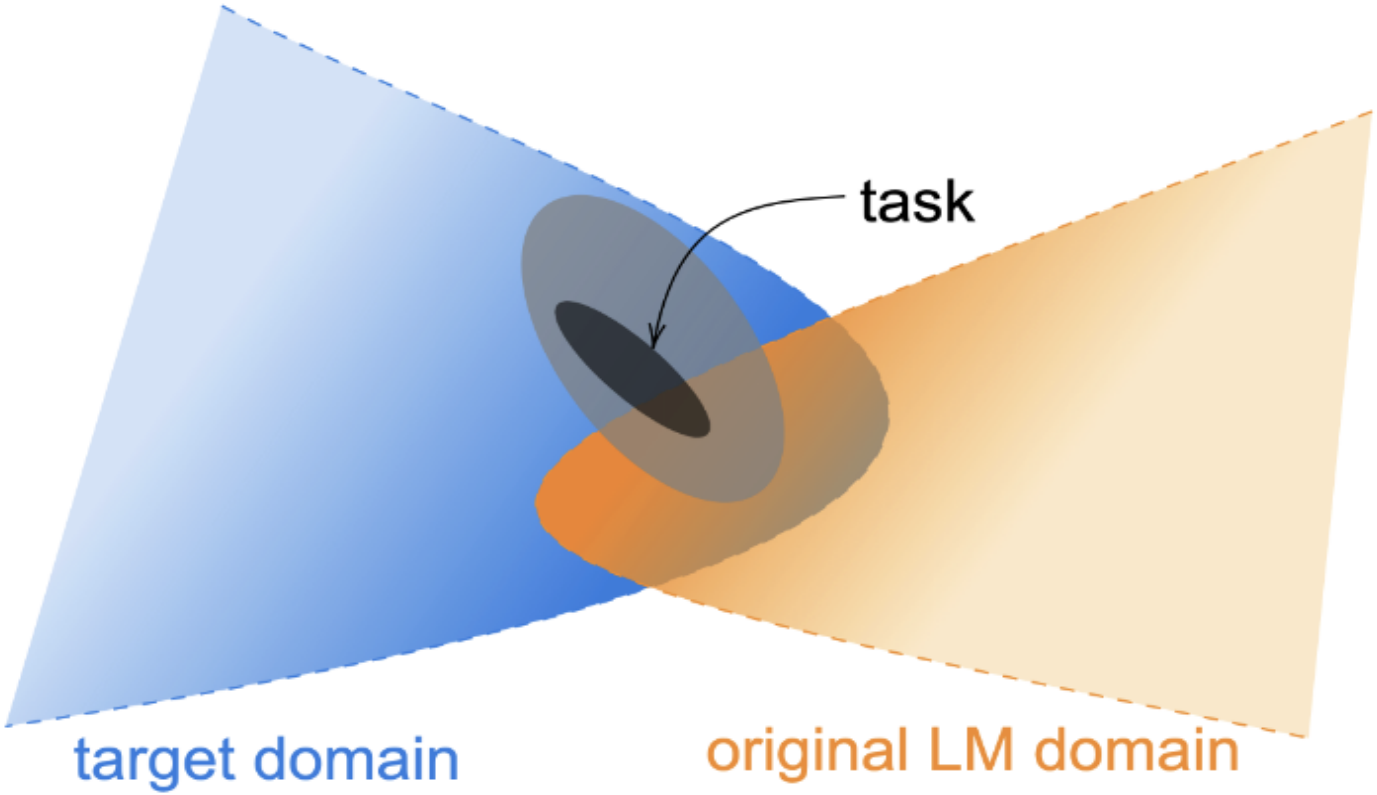
HOW TO SOLVE A TASK WITH LANGUAGE MODELS? (4)

- instruction tuning (short for instruction finetuning)
  - you teach the model a *general* capability of being “helpful”
  - it then solves arbitrary tasks without few-shot prompting and without additional finetuning
  - theory/hope: you no longer have to explicitly train on specific tasks, the instruction-tuned model can solve new tasks without having been trained on them
  - next lecture

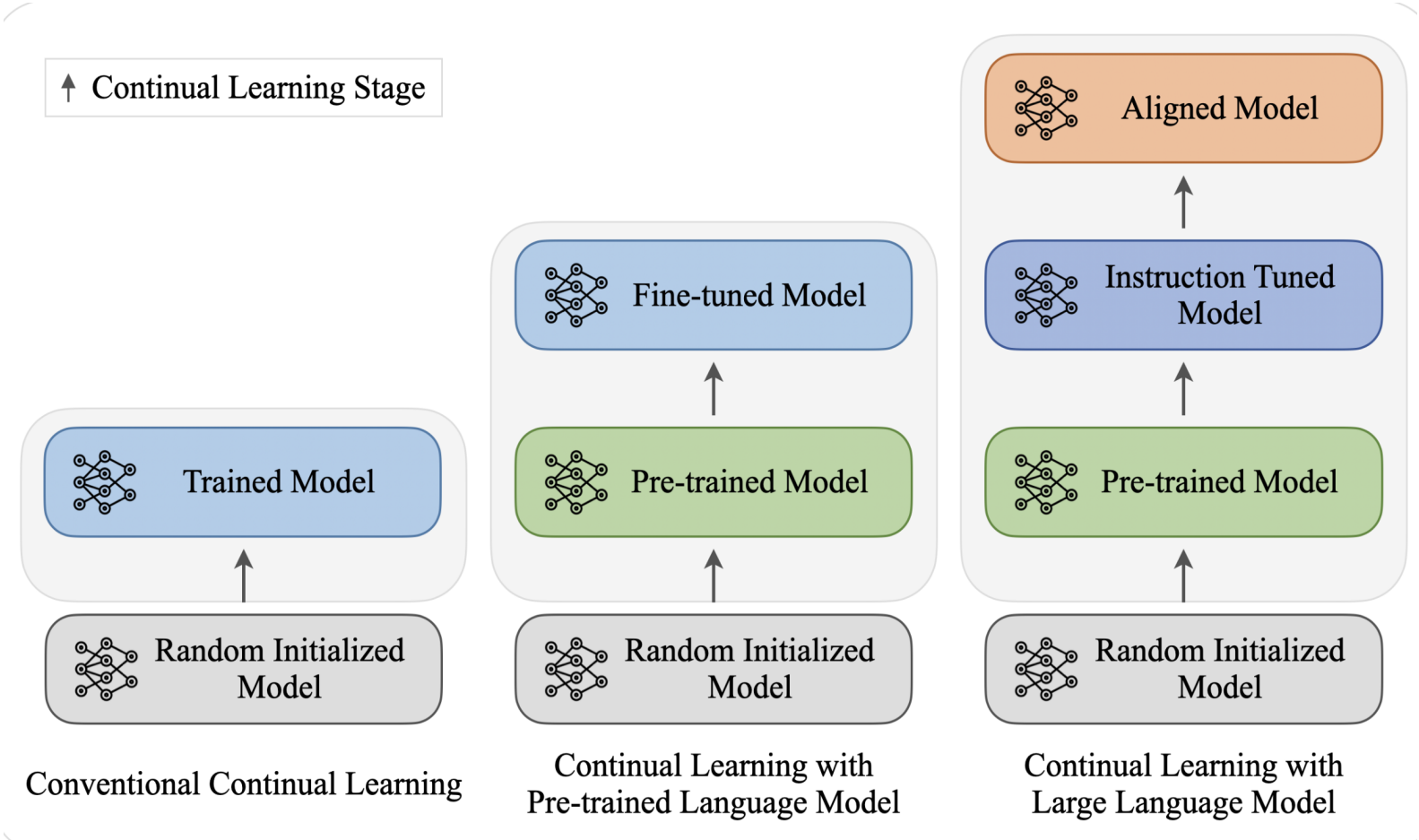
ANOTHER TYPE OF FINETUNING

- Finetuning has yet another meaning.
- Continued pretraining is also sometimes called finetuning.
- Continued pretraining: given a language model that has been trained on generic data (web, reddit etc), adapt it to a new domain (e.g., company-internal data) by training it on a large corpus from this new domain.
- objective: standard language modeling objective
- This results in a language model that has all the nice capabilities of a generic language model (e.g., MISTRAL family), but also understands the special domain.
- Not trivial to do well

CONTINUED PRETRAINING



CONTINUED PRETRAINING



FINETUNING: TERMINOLOGY

- single-task finetuning
- multi-task finetuning
- instruction tuning
  (can be seen as a form of finetuning)
- prompting
  (this is definitely not finetuning)
- continued pretraining
- **Question:** terminology clear?

ISSUES WITH SINGLE-TASK FINE-TUNING

- The result is a single-task model.
- Sequential transfer learning instead of true multi-task learning
- Generalization of the model only w.r.t. to one task / data distribution
- Requires quite a bit of annotated data (e.g., $k = 100$)
- Single-task models have poor few-shot capabilities
- **Question:** Could there be other tasks that might benefit?
- **Question:** What about related domains / languages?

ISSUES WITH (TRADITIONAL) PROMPTING

- Assumption: Model has learned about the task during (unsupervised) pre-training
- Write prompt so that a direct response must be given by the language model.
- Just a label, just yes/no answer, just a name in QA
- This is not natural dialogic behavior: humans typically don't just answer with a label, yes/no, a name (although sometimes they do)
- See next lecture
- Again: Prompting works best if the task has occurred during unsupervised training.
- Current use of prompting is more general: basically anything you type into the LLM context window
- **Question:** For which tasks is this expected to work well?

ISSUES WITH FEW-SHOT PROMPTING: (1) CLASSIFICATION

- Assumption: Model has learned about the task during (unsupervised) pre-training
- Write prompt so that a direct response must be given by the language model.
- Just a label, just yes/no answer, just a name in QA
- This is not natural dialogic behavior: humans typically don't just answer with a label, yes/no, a name (although sometimes they do)
- See next lecture
- Again: Prompting works best if the task has occurred during unsupervised training.
- Current use of prompting is more general: basically anything you type into the LLM context window
- **Question:** For which tasks is this expected to work well?

ISSUES WITH FEW-SHOT PROMPTING: (2) TEXT GENERATION

- Very similar to (1) classification
- Assumption: Model has learned about the task during (unsupervised) pre-training
- Write prompt so that a direct response must be given by the language model.
- Works best if the task has occurred during unsupervised training.
- **Question:** For which tasks is this expected to work well?

MULTI-TASK LEARNING: FLAN AND T0

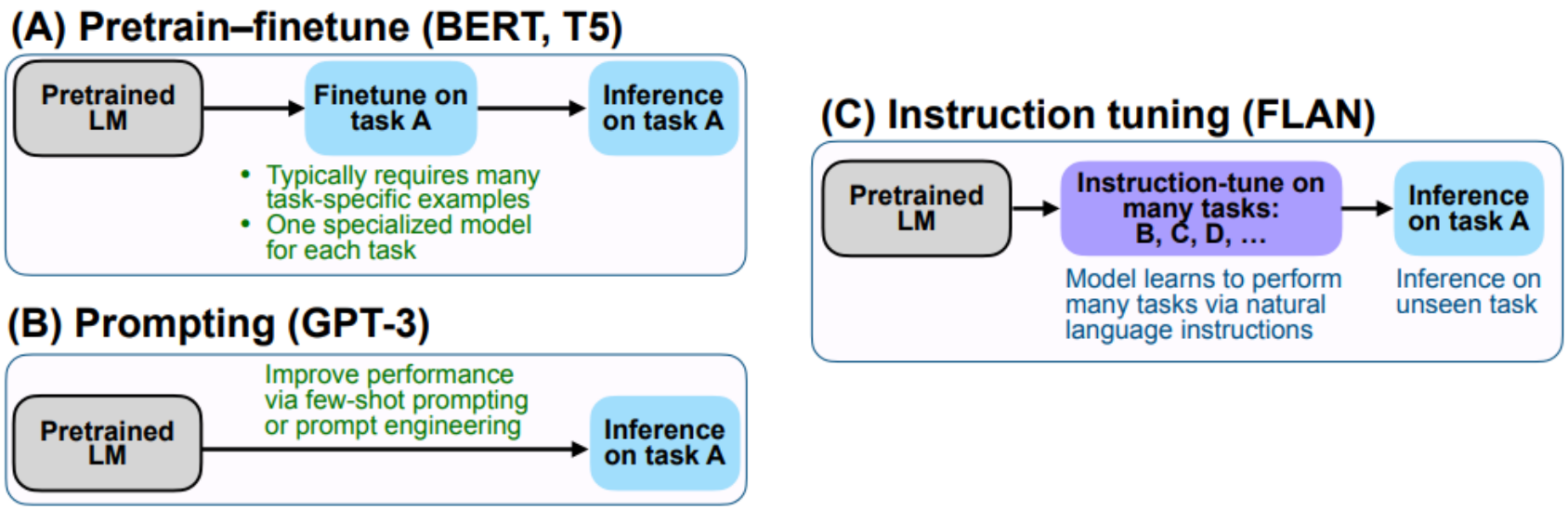
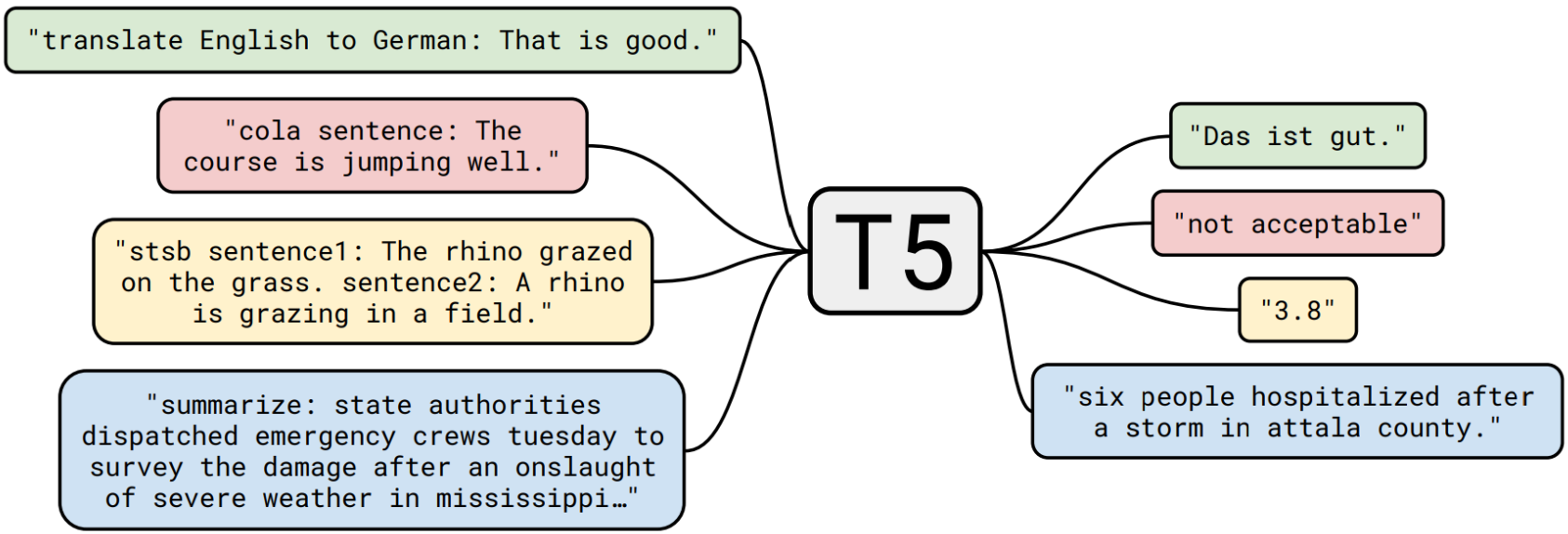


Figure 2: Comparing instruction tuning with pretrain–finetune and prompting.

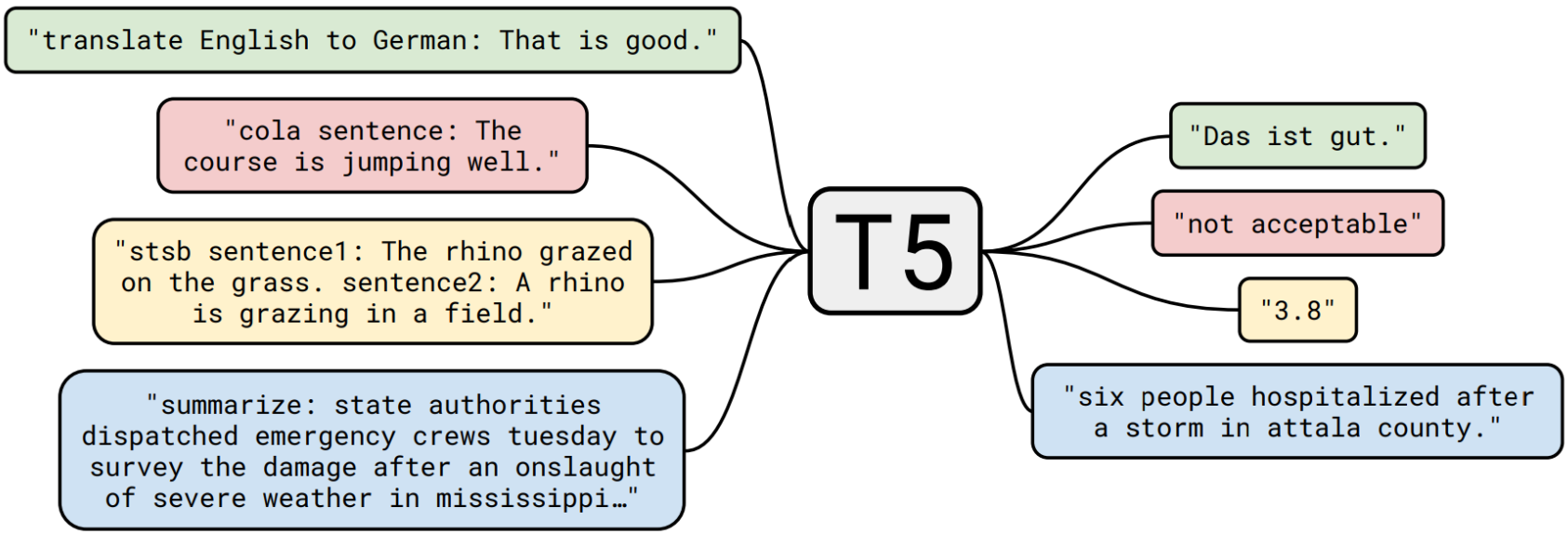
▸ Source: Wei et al., 2021

MULTITASK FINETUNING: BEST OF BOTH WORLDS (FINETUNING AND PROMPTING)



▸ Source: Raffel et al., 2020

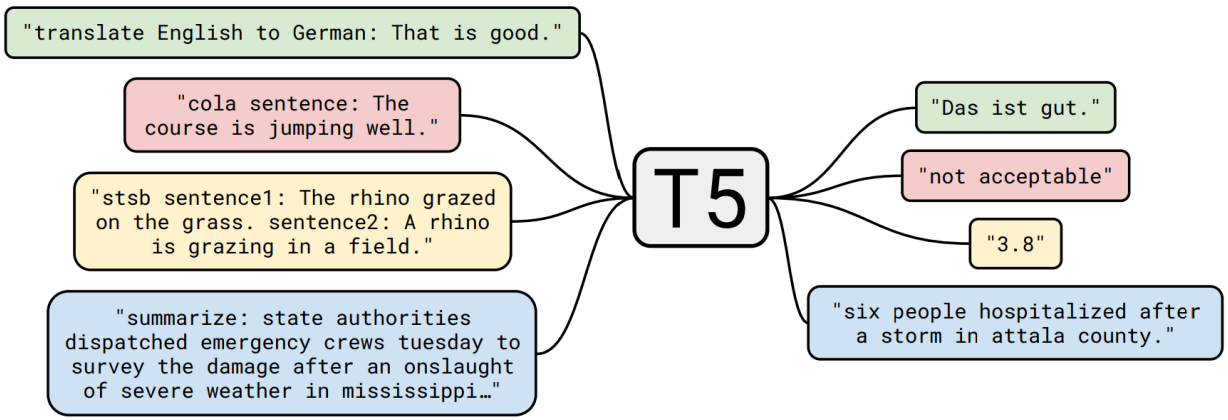
MULTITASK FINETUNING: BEST OF BOTH WORLDS (FINETUNING AND PROMPTING)



▸ Source: Raffel et al., 2020

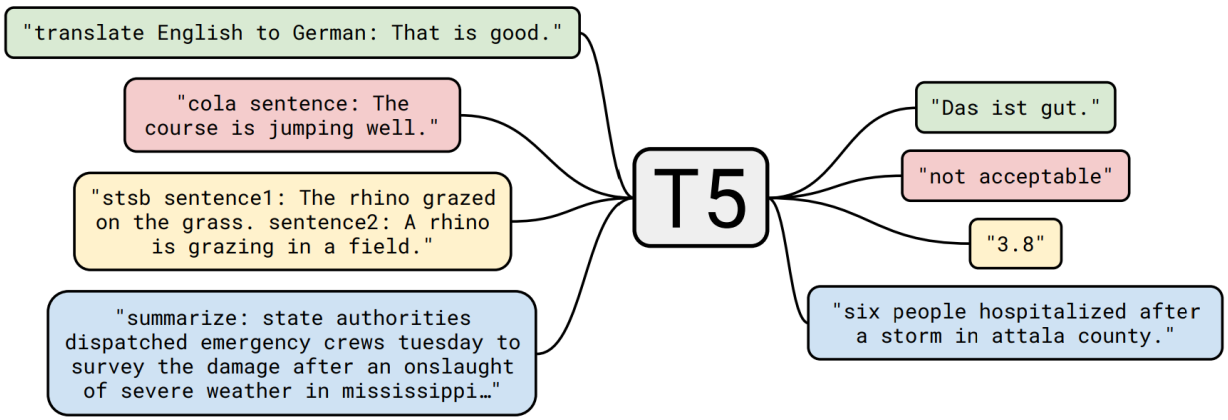
Question: What exactly does the model learn here? How well would we expect this to generalize to new tasks?

CONSTRUCTION OF TRAINING SET



▸ Source: Raffel et al., 2020

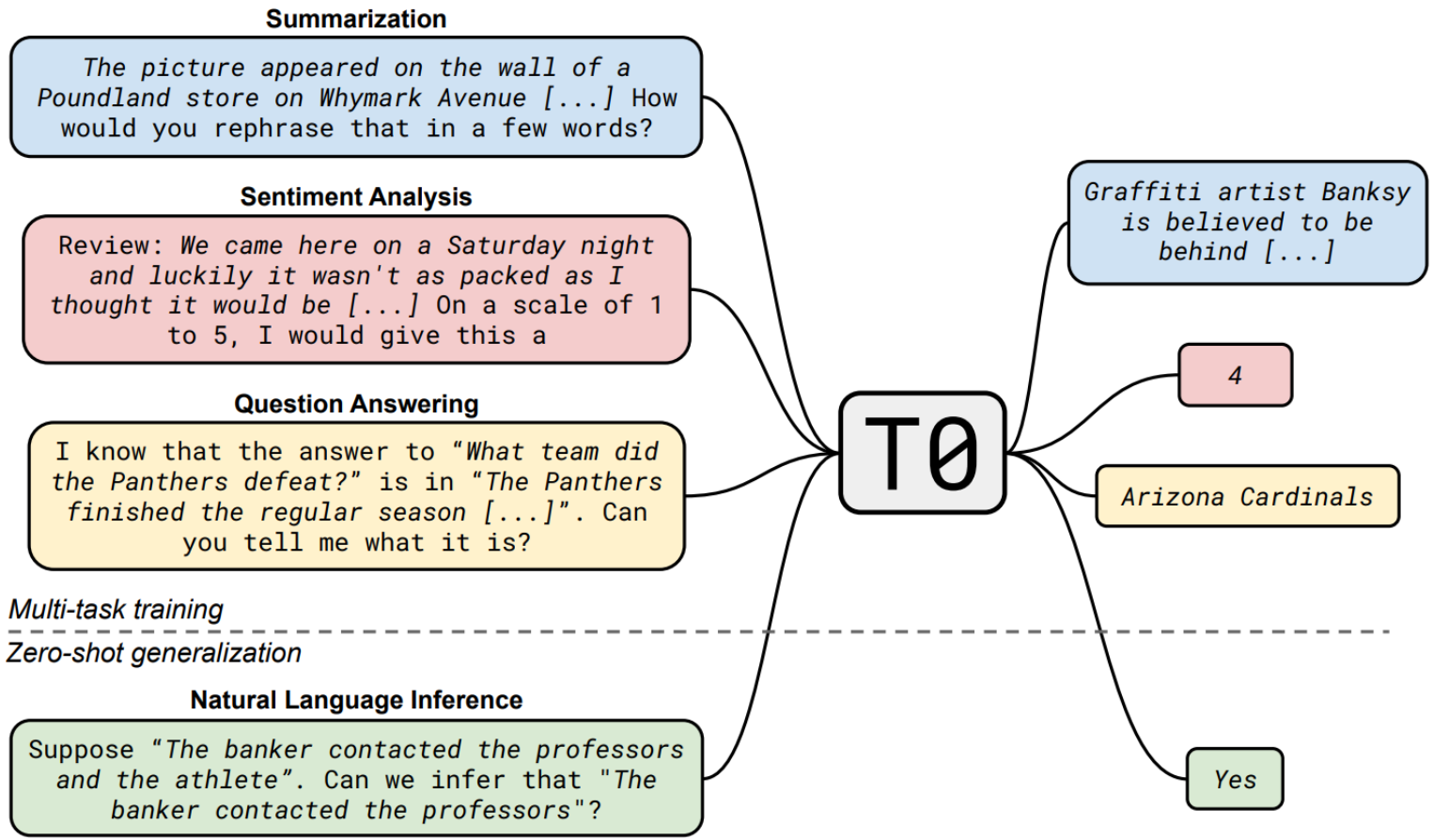
CONSTRUCTION OF TRAINING SET



▸ Source: Raffel et al., 2020

- **Question:** The format used here is not just any format, but it has a special property. Which property is this?

CAREFULLY DESIGNING TASK FORMATS (MOSTLY PREFIXES)



▸ Source: Sanh et al., 2021

T0 AUTHORS CALL THIS “MULTITASK PROMPTED TRAINING”

- *Multitask Prompted Training:* This training method involves learning from multiple tasks using unified prompt formats as a means to improve generalization to new, unseen tasks.
- This means that the model can perform well on tasks it hasn’t been explicitly trained on.
- The key for this lies in the set of shared prompts it has learned from during fine-tuning.

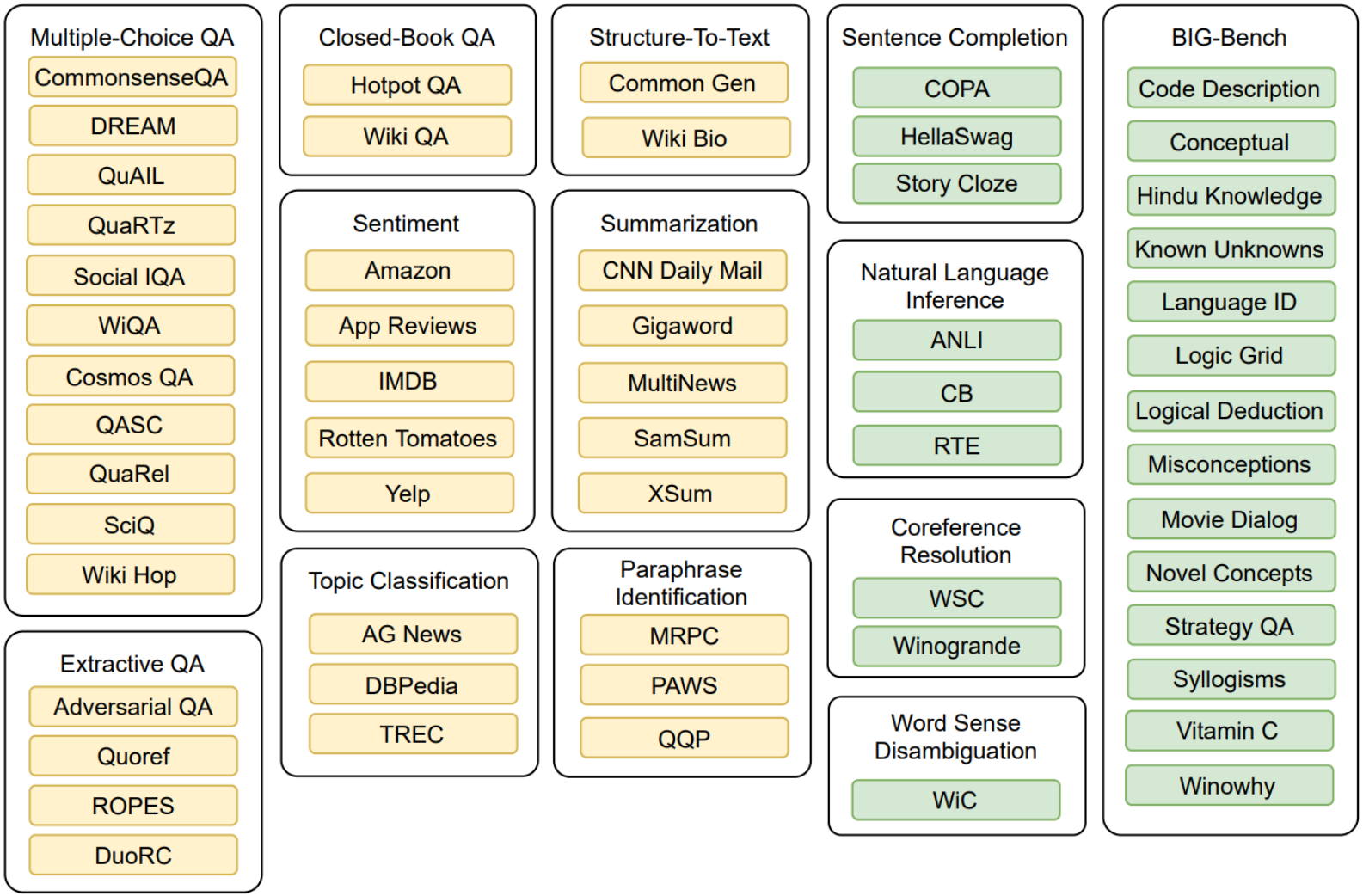
MULTITASK PROMPTED TRAINING

- Benchmark for Evaluation:
  - Instead of using held-out *samples* (as is standard in NLP)
    ...
  - ...we now use held-out *tasks*
  - All data sets belonging to a held-out task go to the test set
  - Generalization across tasks
- Highlights the importance of prompts: The paper emphasizes the importance of prompts in facilitating transfer to new tasks, as the model can generalize to new tasks by relying on the learned prompts and the ability to generate text outputs.

COREFERENCE RESOLUTION

- One of the tasks on the next slide
- In the simplest case the task is to determine which noun phrase a referring expression like “she” or “this person” is referring to.
- Example: “Peter helped Paul because he was familiar with the problem.”
- “he” refers to Peter, not to Paul.

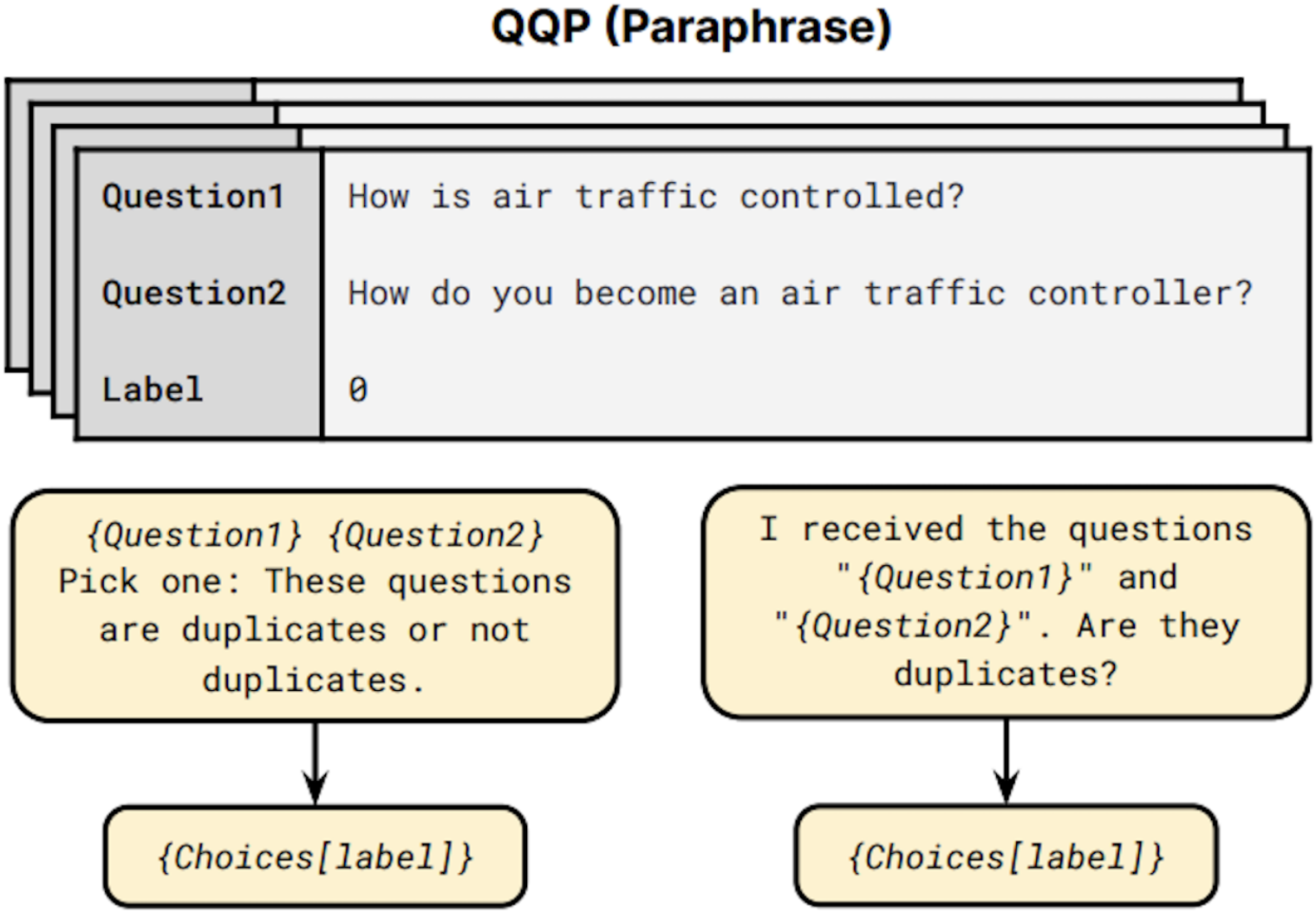
T0: TRAINING AND TEST TASKS



▸ Source: Sanh et al., 2021

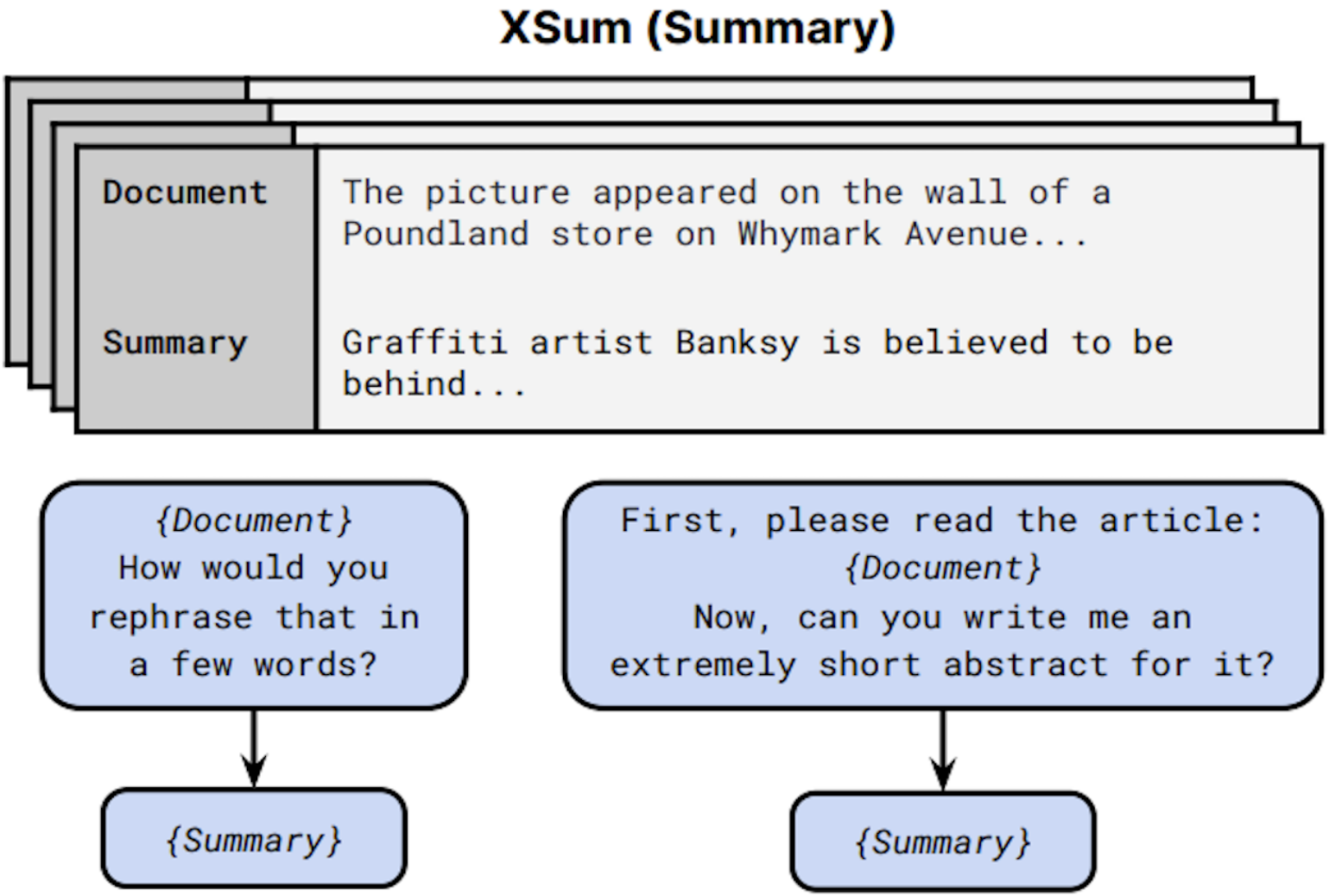
Question: What is an easy task? What is a hard task?

T0 – PROMPT TEMPLATES



▸ Source: Sanh et al., 2021

T0 – PROMPT TEMPLATES



▸ Source: Sanh et al., 2021

FINETUNED LANGUAGE NET (FLAN)

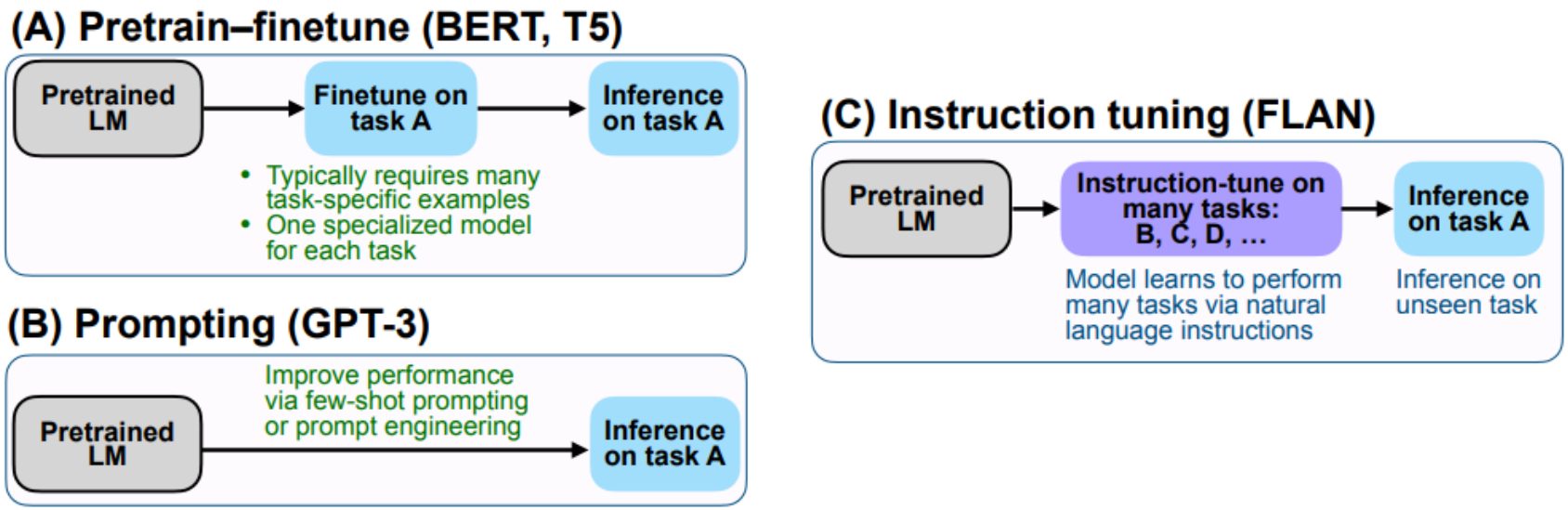
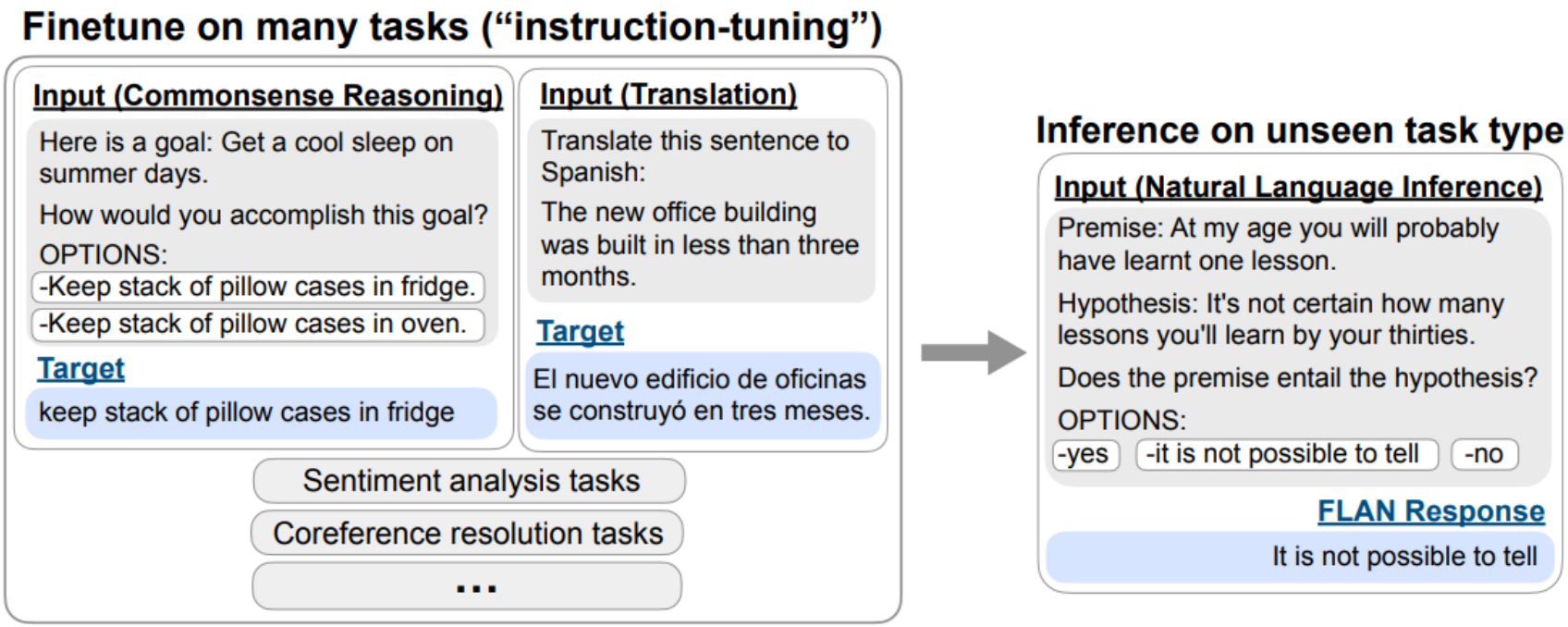


Figure 2: Comparing instruction tuning with pretrain–finetune and prompting.

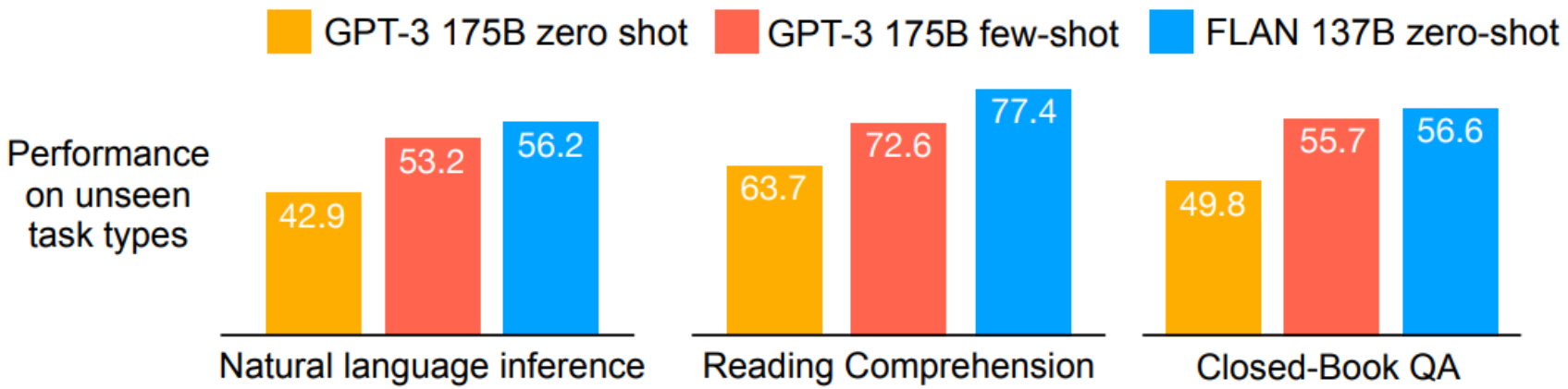
▸ Source: Wei et al., 2021

FLAN FINETUNING



▸ Source: Wei et al., 2021

FLAN PERFORMANCE



▸ Source: Wei et al., 2021

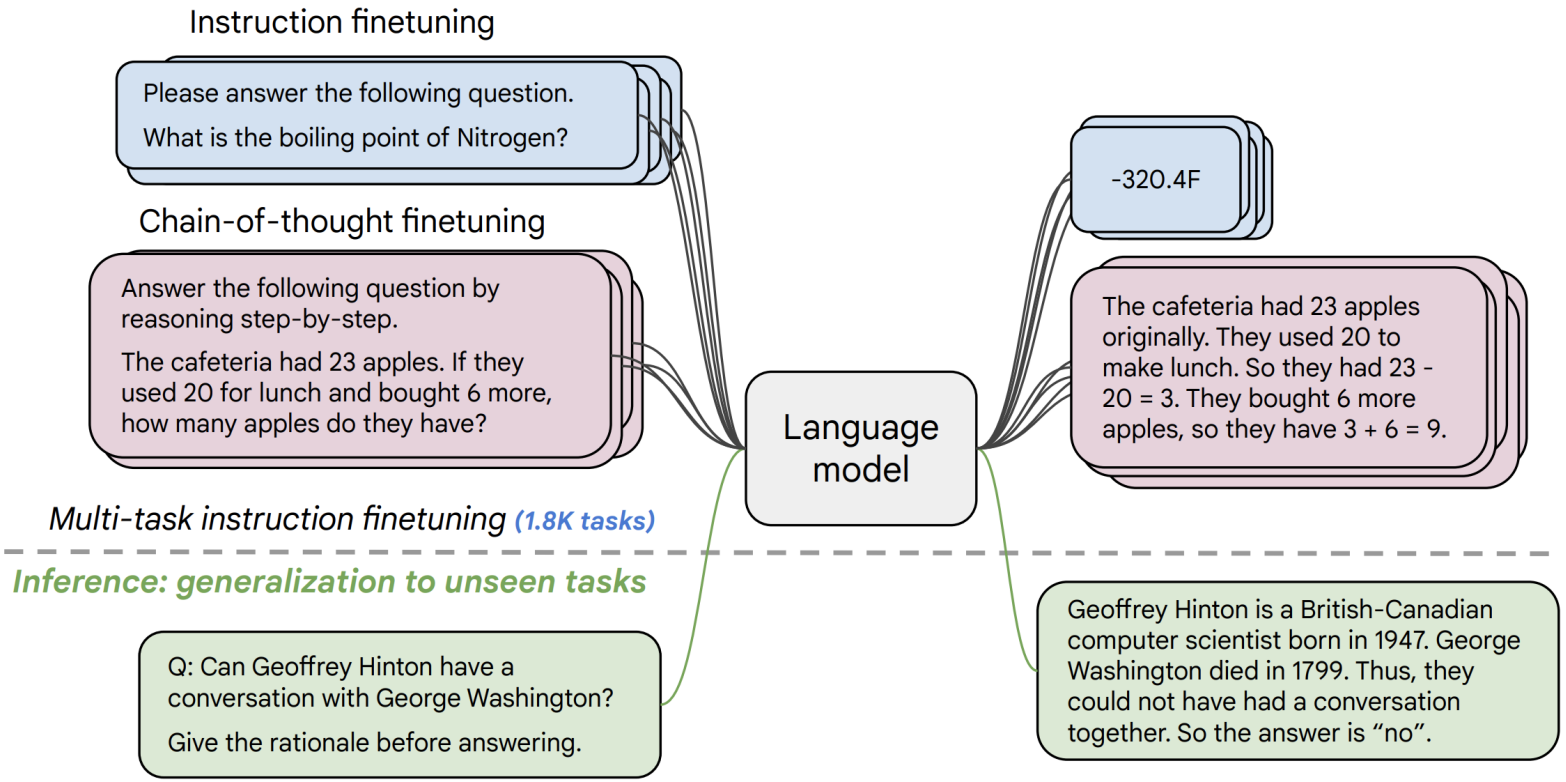
FLAN FINE-TUNING

Extend multi-task learning

- Scaling the number of tasks and data
  - NIV2 (1554 tasks)
  - T0-SF (193 tasks)
  - Muffin (80 tasks)
  - CoT (reasoning tasks, cf. next chapter)
- Scaling model sizes
  - PaLM 8 B
  - PaLM 62 B
  - PaLM 540 B

FLAN UPSCALING

Fine-tuning in 1.8 K tasks



▸ Source: Chung et al., 2022

MULTI-TASK TRAINING VS INSTRUCTION TUNING

- The term “instruction tuning” was introduced by FLAN paper,
  - Source: Wei et al., 2021
- However, today “instruction tuning” refers to a wide variety of methods that train on instruction-output pairs, not just tasks
- Open-ended generation and “alignment” is now included under the rubric “instruction tuning”: brainstorming, writing a joke, give me ten analogies for concept X, don’t use hate language, don’t take positions on controversial political issues etc
- These are not classical NLP tasks.
- See next lecture
- Our use of terminology in this class:
- multi-task finetuning = tasks in the classical NLP sense
- instruction-tuning: a broad approach to changing the behavior of a raw language model (i.e., trained on next-word prediction) to be “helpful, harmless and honest”. This includes training on classical NLP tasks, but it’s just one part.

FINE-TUNING CONCLUSIONS

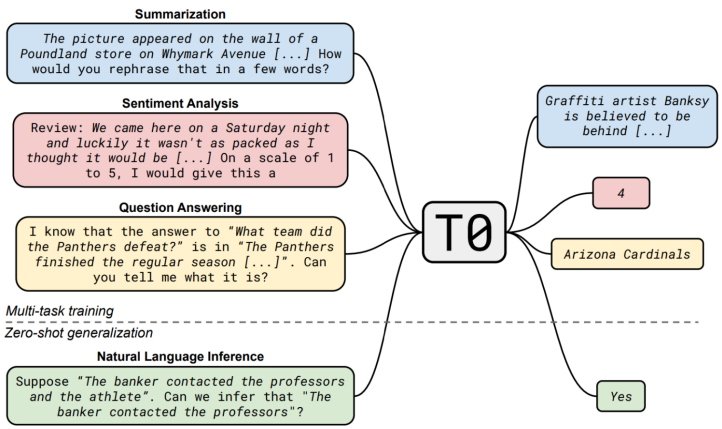
- The more the better
  - The more parameters the better
  - The more training tasks the better
- Multi-task finetuning generalizes across models
  - It works well on different architectures
- Greatly improves usability
- It is relatively compute-efficient
  - Pretraining is hugely expensive, finetuning and instruction tuning are usually much cheaper.
  - For PaLM 540 B it takes 0.2 % of pre-training compute, but improves by 9.4 %

SLIDO

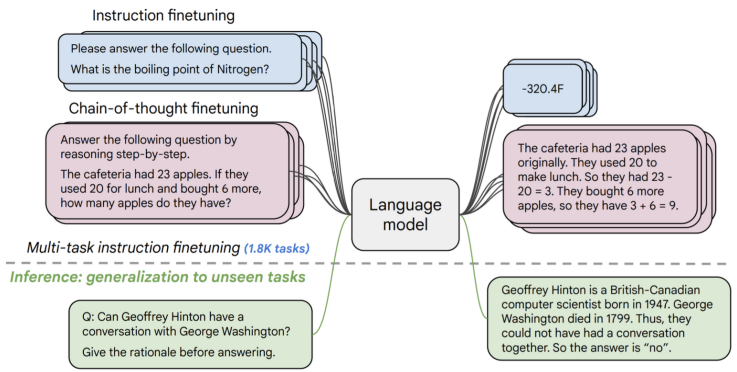
-
- slido
- #1312409

T0 VS FLAN: DIFFERENCE?

CAREFULLY DESIGNING TASK FORMATS (MOSTLY PREFIXES)



Fine-tuning in 1.8 K tasks



PROMPTING: DEFINITION

- Prompting always means that you do not finetune the model, i.e., you don't change the parameters of the model.
- The term prompting is generally used when you carefully design the input to the language model, so that you will get the desired output.
- But nowadays any type of input to the model can be called a prompt.
- Few-shot or k-shot prompting means that you give a few examples or k examples of what you want the model to do as part of the input to the model.
- Zero-shot prompting means that you do not give an example. The term zero-shot prompting is often used in the context of few-shot prompting (contrasting using $k > 0$ examples vs 0 examples).

HOW TO LLMS DO MATH, E.G., ADDITION?

ARITHMETIC WITHOUT ALGORITHMS: LANGUAGE MODELS SOLVE MATH WITH A BAG OF HEURISTICS

Yaniv Nikankin[1*] **Anja Reusch**[1] **Aaron Mueller**[1,2] **Yonatan Belinkov**[1]
[1]Technion – Israel Institute of Technology [2]Northeastern University

ABSTRACT

Do large language models (LLMs) solve reasoning tasks by learning robust generalizable algorithms, or do they memorize training data? To investigate this question, we use arithmetic reasoning as a representative task. Using causal analysis, we identify a subset of the model (a circuit) that explains most of the model's behavior for basic arithmetic logic and examine its functionality. By zooming in on the level of individual circuit neurons, we discover a sparse set of important neurons that implement simple heuristics. Each heuristic identifies a numerical input pattern and outputs corresponding answers. We hypothesize that the combination of these heuristic neurons is the mechanism used to produce correct arithmetic answers. To test this, we categorize each neuron into several heuristic types—such as neurons that activate when an operand falls within a certain range—and find that the unordered combination of these heuristic types is the mechanism that explains most of the model's accuracy on arithmetic prompts. Finally, we demonstrate that this mechanism appears as the main source of arithmetic accuracy early in training. Overall, our experimental results across several LLMs show that LLMs perform arithmetic using neither robust algorithms nor memorization; rather, they rely on a "*bag of heuristics*". [1]