Large Language Models (LLMs)

Fine-Tuning

Learning goals

- comprehend the different subtleties in the space between supervised fine-tuning and zero-shot prompting

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

- "old-style" single-task fine-tuning
  - supervised training 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), typically done for BERT
  - alternatively, the output can be a meaningful "verbalizer" (e.g., "negative", "positive" for sentiment analysis), works best for autoregressive models

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

- few-shot prompting
  - provide k (where k is small) 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

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

- multi-task finetuning
  - mixes single-task finetuning with prompting
  - requires finetuning on a large training set
  - but now: many tasks (the more the better)
  - task format is consistent with language model objective (similar to verbalizer)
  - T5

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

- instruction tuning (short for instruction finetuning)
  - theory/hope: you do not have to explicitly train on specific tasks
  - instead, you teach the model a general capability of being "helpful"
  - it then solves arbitrary tasks without few-shot prompting and without finetuning beyond instruction tuning
  - no task-specific training set required!
  - 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-iternal data) by training it on a large corpus from this new domain.
- 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.

RECAP

- encoder vs. decoder models (discriminative vs. generative)
- (in-context) learning vs. creative generation/chat
- pre-training with plain language modeling:
  - next token prediction
  - no explicit understanding of tasks
  - no alignment with human preferences for chat or for generation of coherent text

ISSUES WITH (BERT TASK-SPECIFIC) FINE-TUNING

- Only single-task models (sequential transfer learning instead of multi-task learning)
- Generalization of the model
  - only w.r.t. to one task / data distribution
  - **Question:** what about other tasks? Do they also benefit?
  - **Question:** what about related domains? other languages?
- Still requires (quite) large amounts of annotated data
- Poor (to none) zero-/few-shot capabilities of fine-tuned models

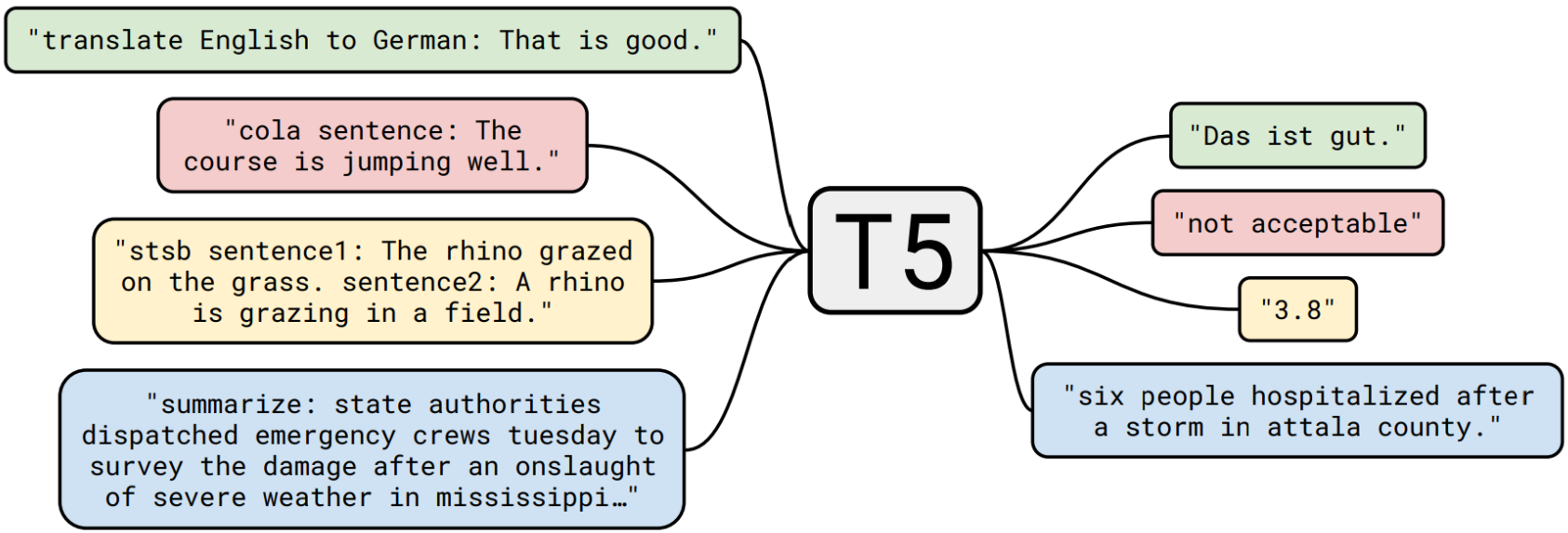
ISSUES WITH PROMPTING

- Assumption: Model has learned about the task during (unsupervised) pre-training
  Question: Is this *always* a realistic assumption??
- A direct response within the frame of a given label set is expected
  - Humans usually don't directly answer but provide intermediate reasoning steps (so-called "chain-of-thought")
- *Misalignment with human needs*
  - Out of context answers
  - Harmful answers
- *Lack of interpretability*
  - Just the answer w/o explanation
  - Big concern about LLMs in general

ISSUES WITH PROMPTING

- *Hallucinations:* Output that is not true. (different possible causes: just made up, incorrect internal reasoning etc)
- *Imprecise mathematical operations:* Models not trained to do arithmetics
- *Inadequate experience grounding:* Not fully capable of generating correct answers to questions from specialized domains not covered by pretraining data
- *Limited ability for complex reasoning:* Long-known challenge in NLP/LLMs

T5: BEST OF BOTH WORLDS, FINE-TUNING ON TASKS AND PROMPTING



▸ Source: Raffel et al., 2020

BEST OF BOTH WORLDS

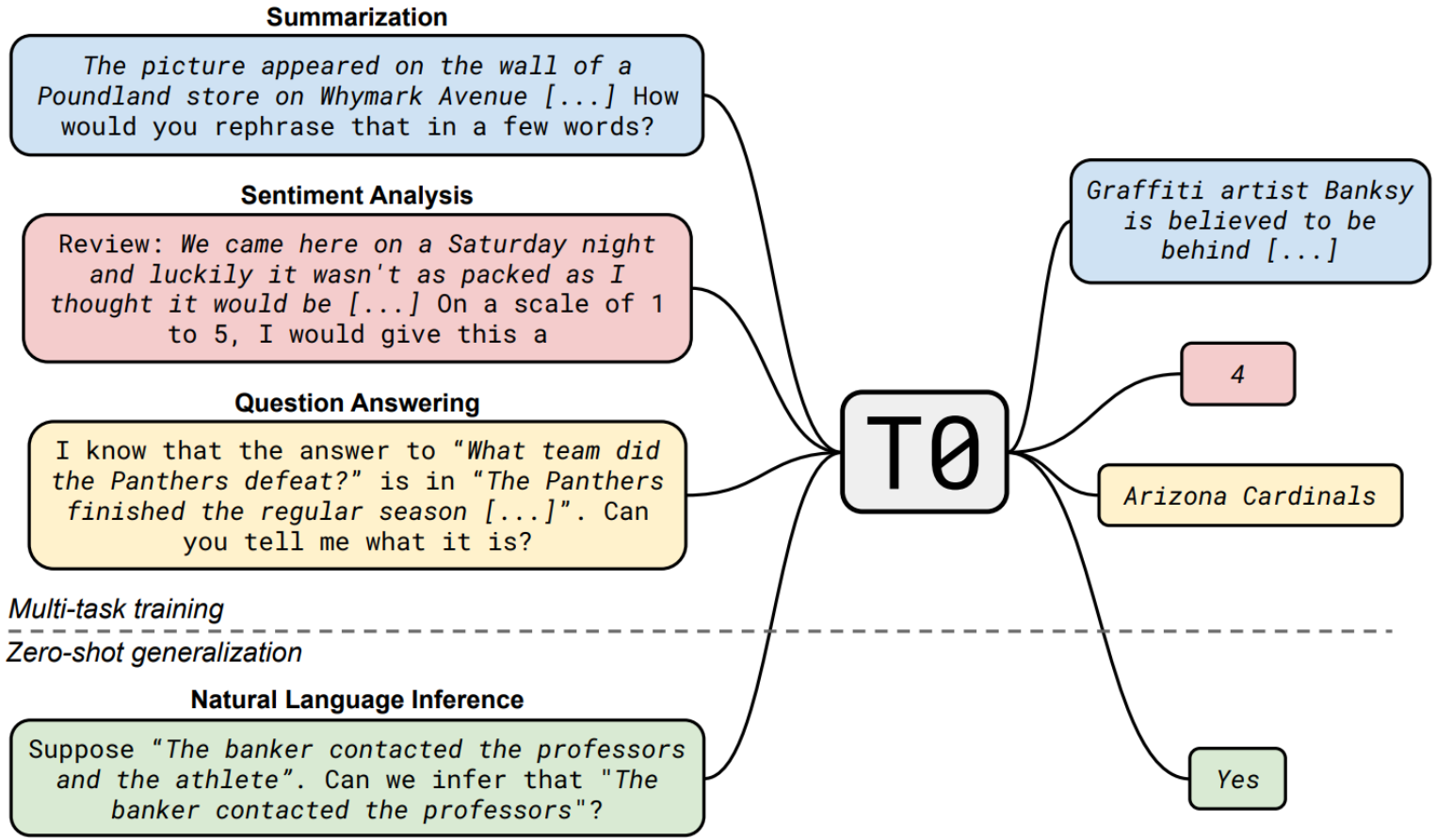
- **Question: How does multi-task learning happen?**

→ IMPLICITLY, i.e. the model learns via fine-tuning which task prefix to associate with which set of labels

- **Question:** How can we make the *EXPLICIT*?

→ Mapping any natural language tasks into a *human-readable* prompted form ▸ Sanh et al., 2021

CAREFULLY DESIGNING TASK PREFIXES



▸ Source: Sanh et al., 2021

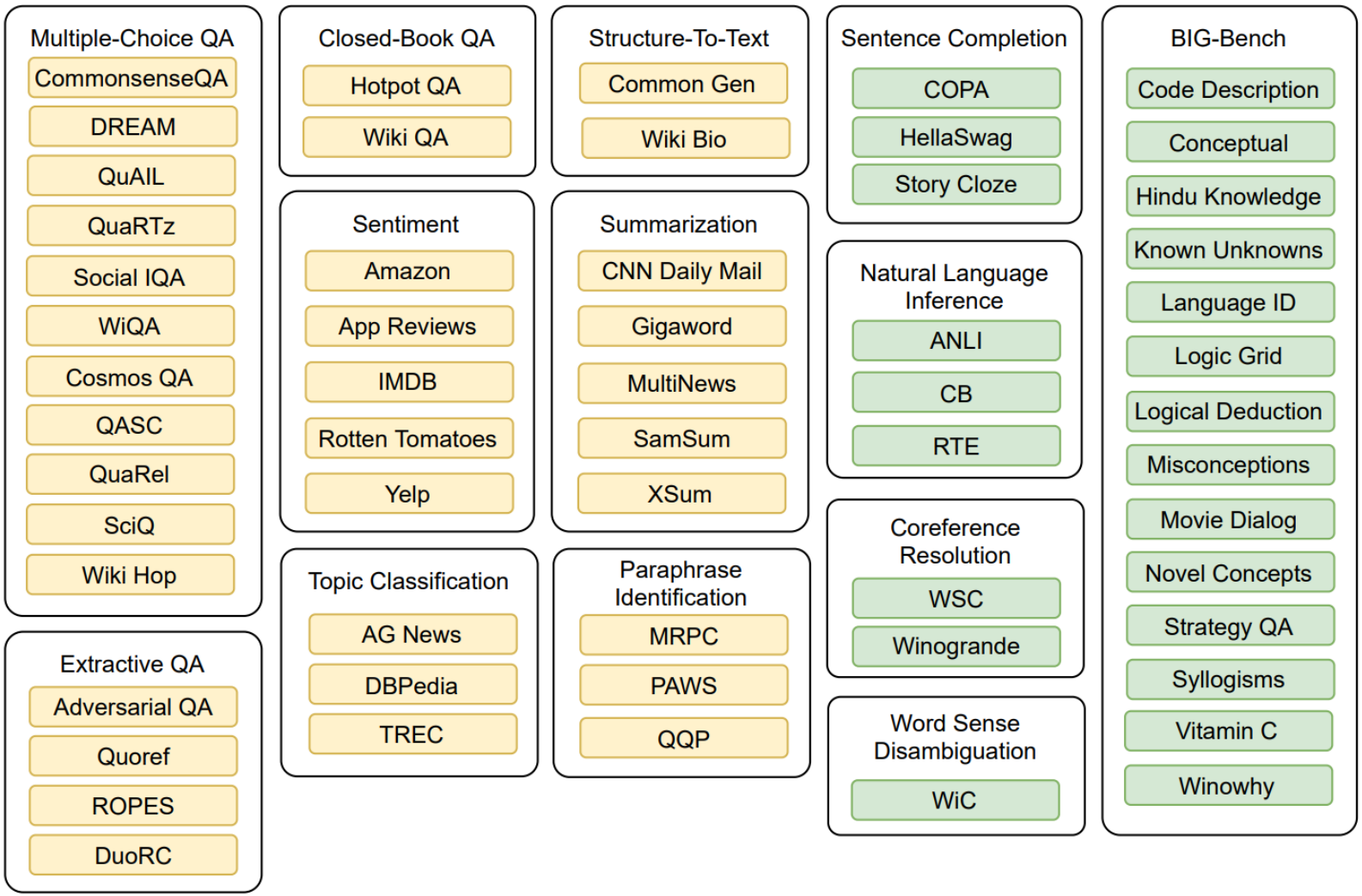
MULTITASK PROMPTED TRAINING

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

MULTITASK PROMPTED TRAINING

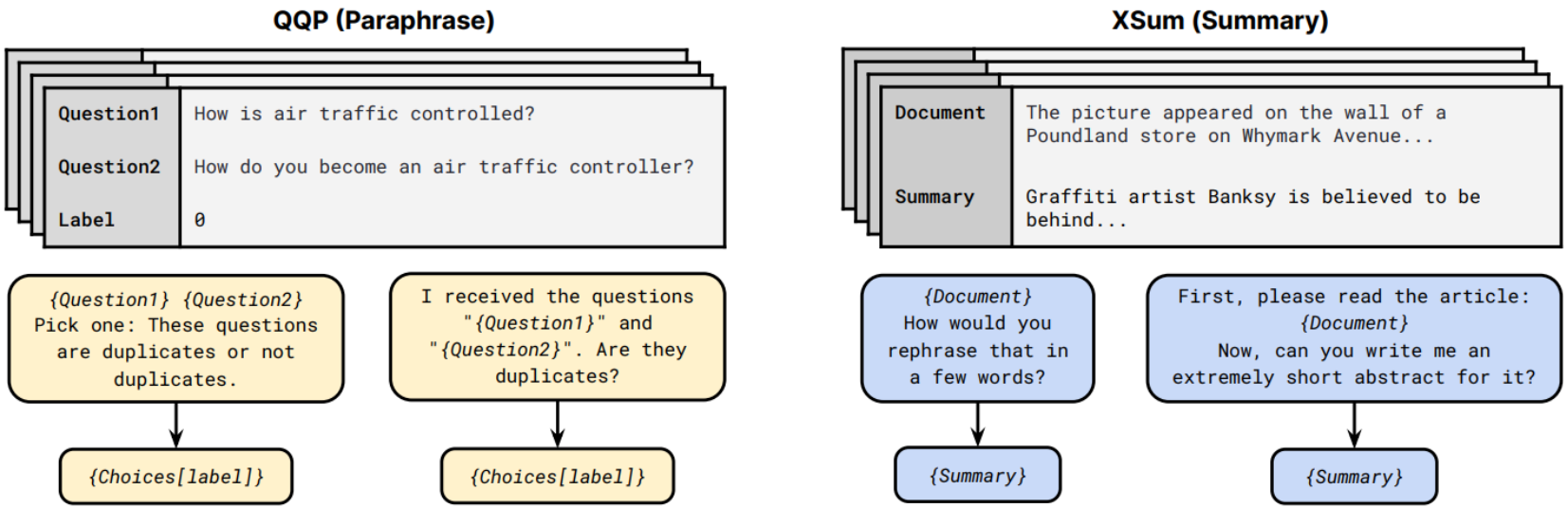
- Benchmark for Evaluation:
  - Held-out *tasks* instead of just held-out samples as a test set (data sets are grouped according to task beforehand)
  - All data sets belonging to as held-out task go to the test set
  - Generalization across tasks / data distributions
- Highlights the Importance of Prompts: The paper emphasizes the importance of prompts in facilitating zero-shot learning, as the model can generalize to new tasks by relying on the learned prompts and the ability to generate text outputs.

T0 – DATA SPLITS



▸ Source: Sanh et al., 2021

T0 – PROMPT TEMPLATES



▸ Source: Sanh et al., 2021

FINETUNED LANGUAGE NET (FLAN)

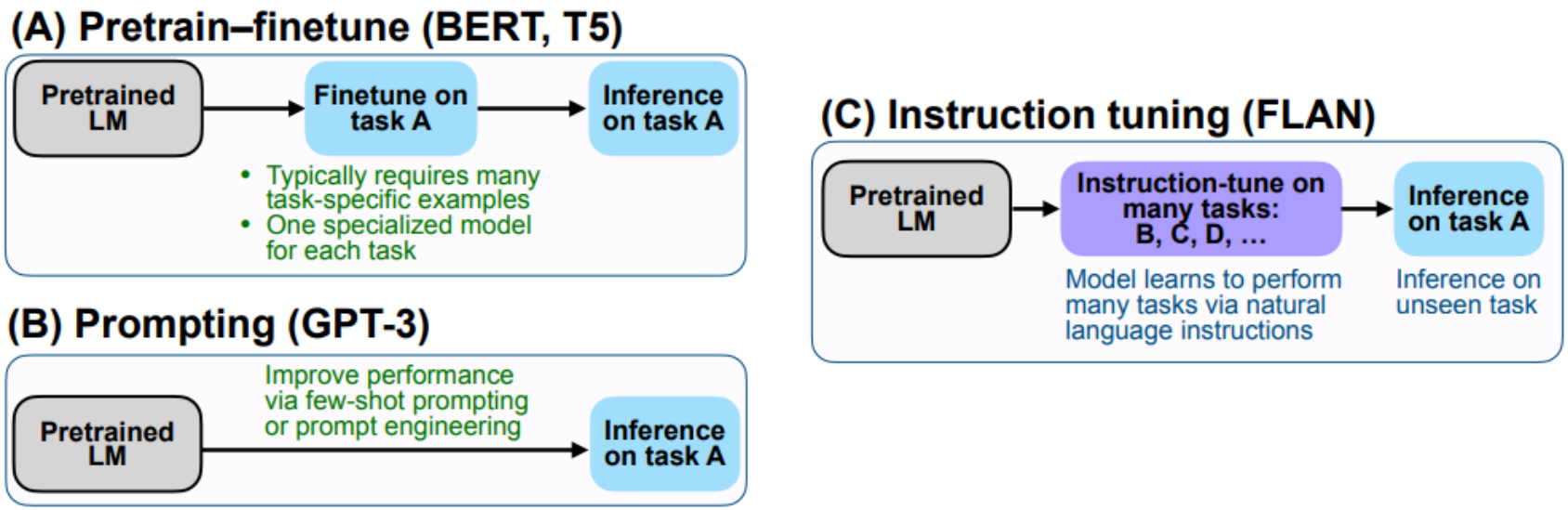
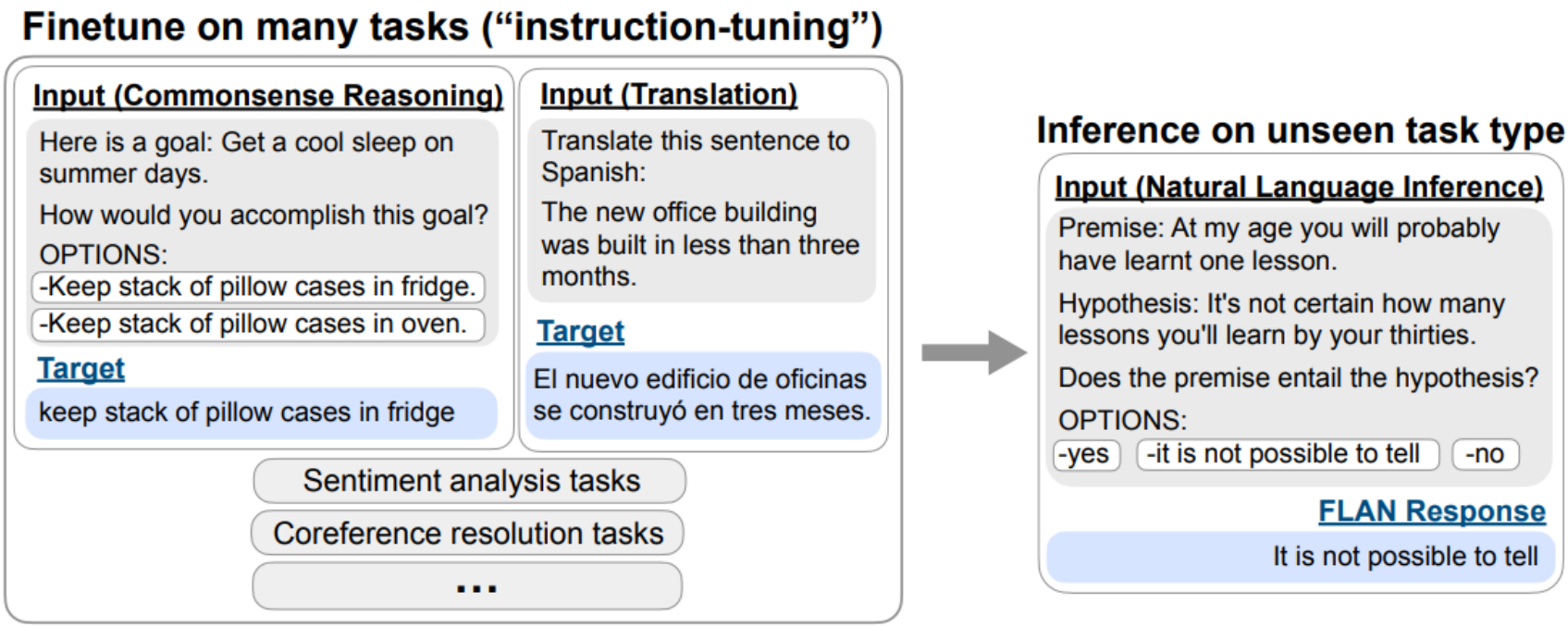


Figure 2: Comparing instruction tuning with pretrain–finetune and prompting.

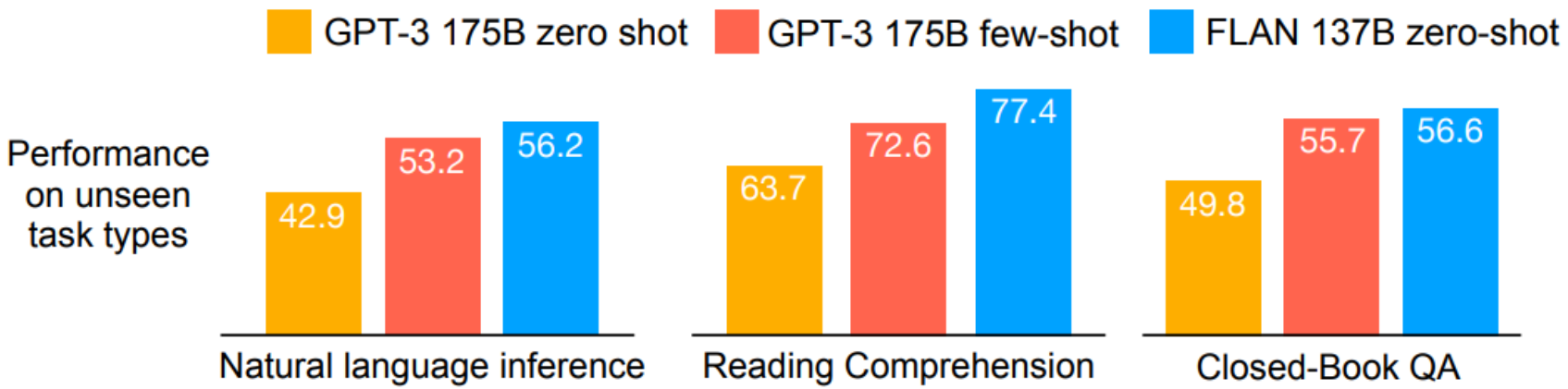
▸ Source: Wei et al., 2021

FLAN FINETUNING



▸ Source: Wei et al., 2021

FLAN PERFORMANCE



▸ Source: Wei et al., 2021

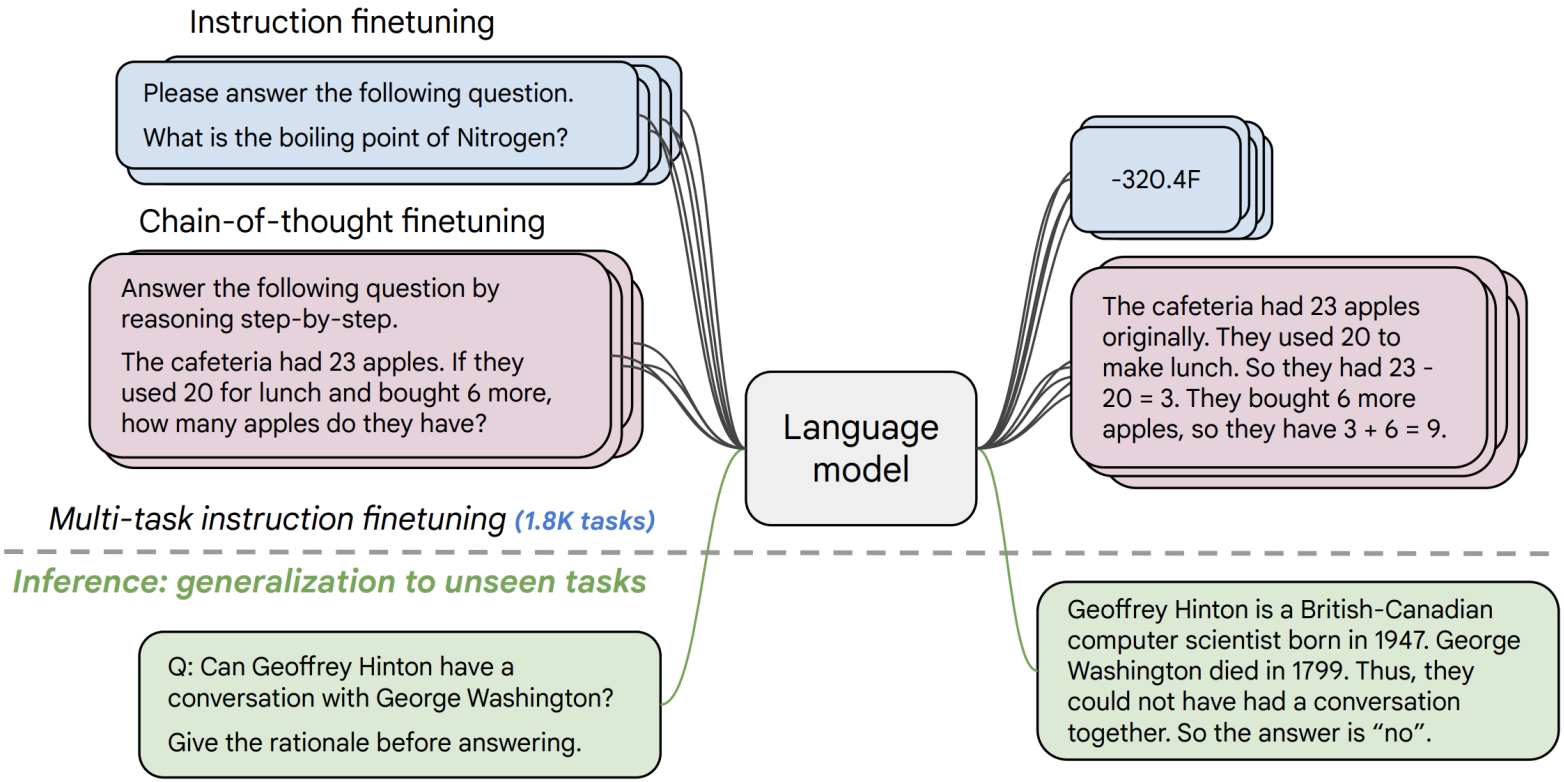
FLAN FINE-TUNING

Extend instruction fine-tuning:

- Scaling the number of fine-tuning 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

FINE-TUNING CONCLUSIONS

- It is still possible to upscale
  - Larger models will improve performance
  - More fine-tuning tasks will improve performance
- Instruction finetuning generalizes across models
  - It works well on different architectures
- It improves usability and mitigates some harms
- It is relatively compute-efficient
  - For PaLM 540 B it takes 0.2 % of pre-training compute, but improves by 9.4 %