Multi-Task Instruction Tuning for LLMs

Sebastian Wiktorowicz, Tao Groves, Andy Phan

02/25/2026

Presentation overview

01

Is it possible for LLMs to perform well at tasks they have never seen before?

Fine Tuned Language Models Are Zero-Shot Learners (FLAN)

02

What design considerations make this possible?

Cross-Task Generalization via Natural Language Crowdsourcing Instructions

03

How does data and model architecture influence performance?

SUPER-NATURAL-INSTRUCTIONS: Generalization via Declarative Instructions on 1600+ NLP Tasks

Fine Tuned Language Models Are Zero-Shot Learners (FLAN)

Google Research
ICLR 2022

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

Motivation: Zero-Shot vs Few-Shot vs Many-Shot

- Zero-shot: Model performs a task using only natural language **instruction** (no examples of task)
- Few-shot: Model given a **few examples** of input-output pairs in the prompt before inferring on a new example
- Many-shot: Higher scale of few-shot

Background

- Pretrain-finetune = one model per task + no generalization
- Prompting = few-shot + no weight updates
- Instruction Tuning = generalization across (unseen) tasks + weight updates

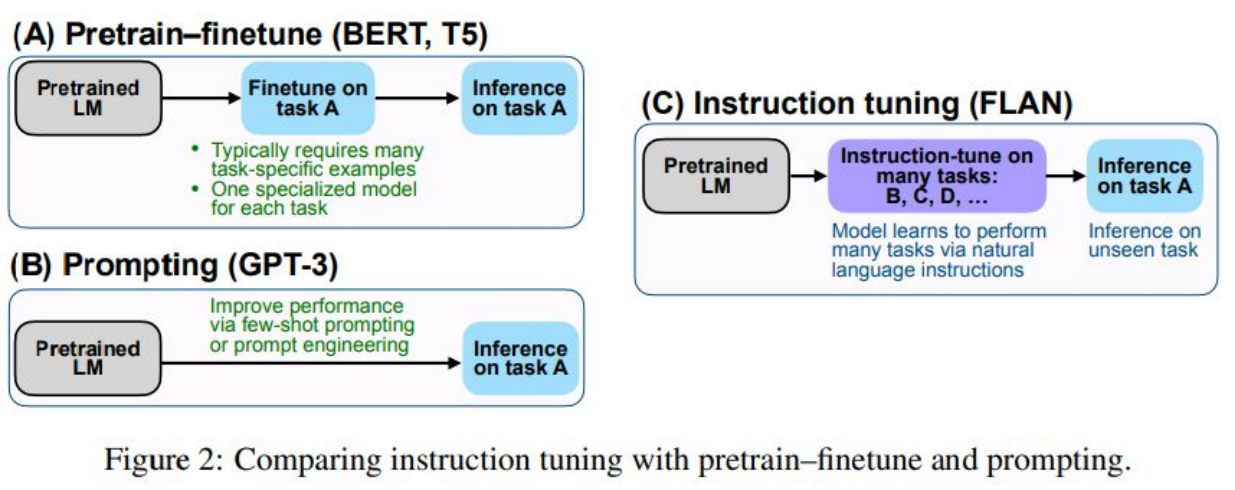
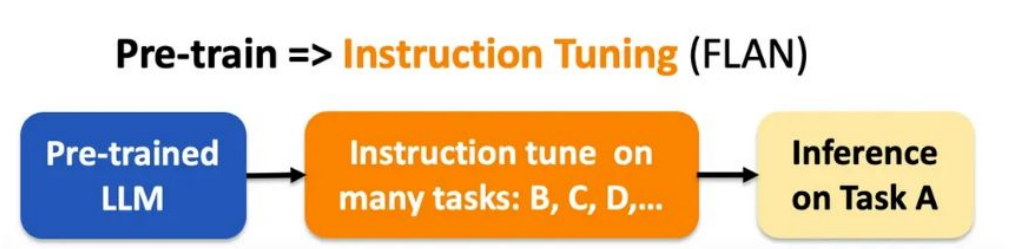


Figure 2: Comparing instruction tuning with pretrain–finetune and prompting.

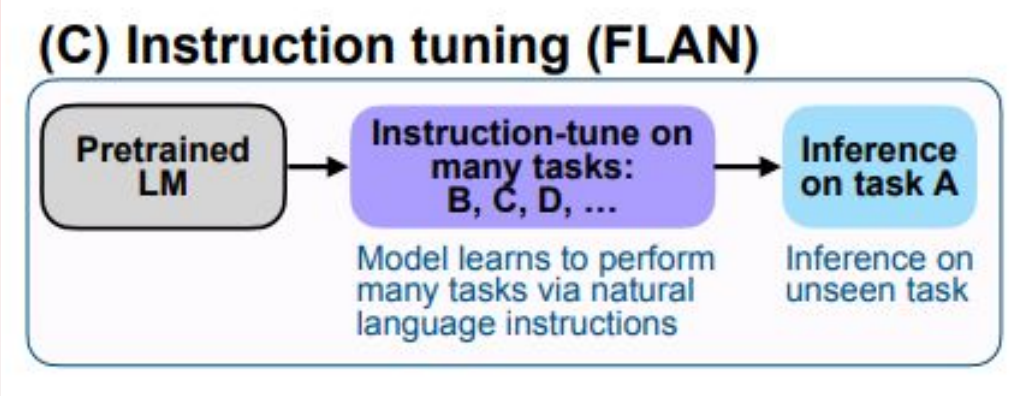
Training Zero-Shot: Is It Possible?

- The methods we saw on the previous slide require k-shot examples before an inference
- Are we able to generalize a model to make inferences without examples and without fine tuning?
- This is called **instruction tuning**



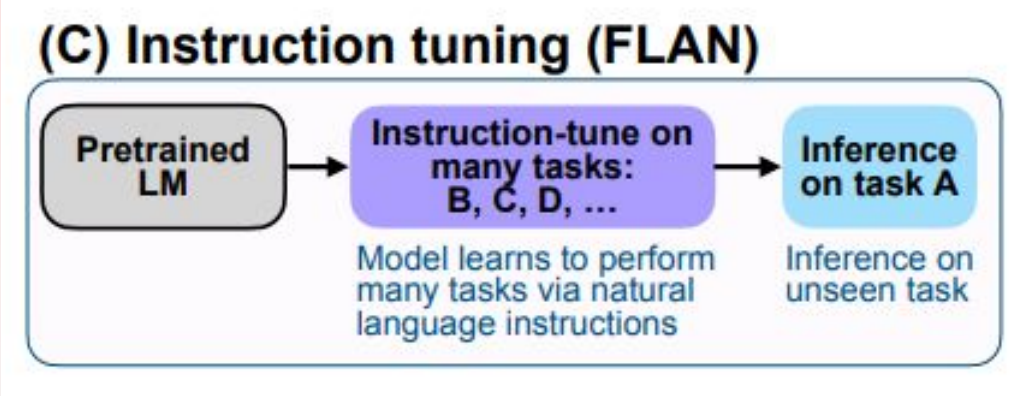
Instruction Tuning

- Instead of tuning a model on a task, we tune **instructions** of a model to do an arbitrary task
- Convert tasks to instructions:
  - “Is the sentiment of this movie review positive or negative?”
  - “Translate ‘how are you’ into Chinese.



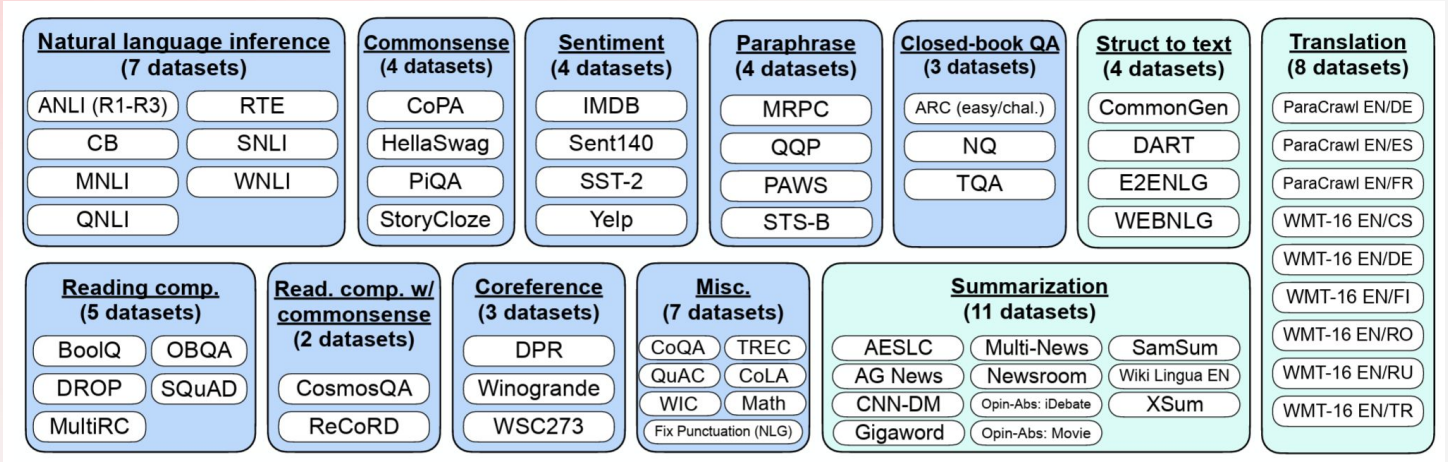
Instruction Tuning

- Instruction tuning can allow a model to understand how to do an arbitrary task without having context examples or seeing the task before
  - Simply by understanding **instructions**
- This is called zero-shot



Datasets + Task Clusters

- 62 datasets
- 12 task clusters
  - Different types of instructions that will be fed into the model during training
  - Variety is important so the model can learn how different instructions are given



Instruction Tuning - Templates

- NLP tasks can be described via natural language instructions
- Done by dividing into 3 separate parts:
  - Premise, hypothesis, options
- 10 templates given for each dataset:
  - Diversity of instruction phrasing

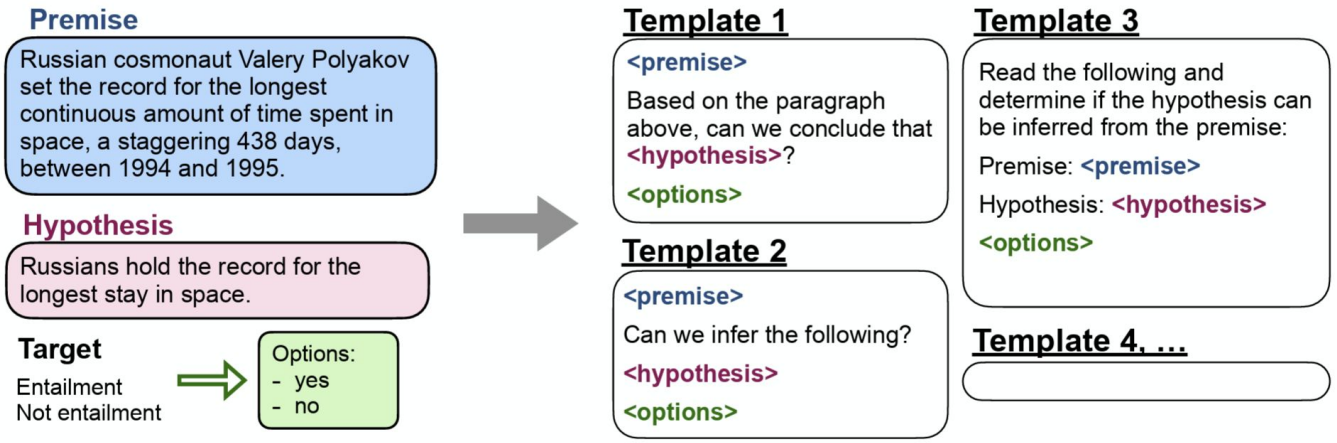


Figure 4: Multiple instruction templates describing a natural language inference task.

Experiment Setup

- Train on all clusters except one (holdout cluster)
- Evaluate zero-shot on this cluster
  - What this allows is for the model to be conditioned/trained on all other clusters and to be evaluated on the test cluster purely without any examples
- Called task generalization:
  - Learn instructions for other tasks to infer on another task

Outputs

Classification output

Responds in free-text, so must get creative

Use **rank classification** - only consider 2 outputs for the model (ex: “yes” and “no”)

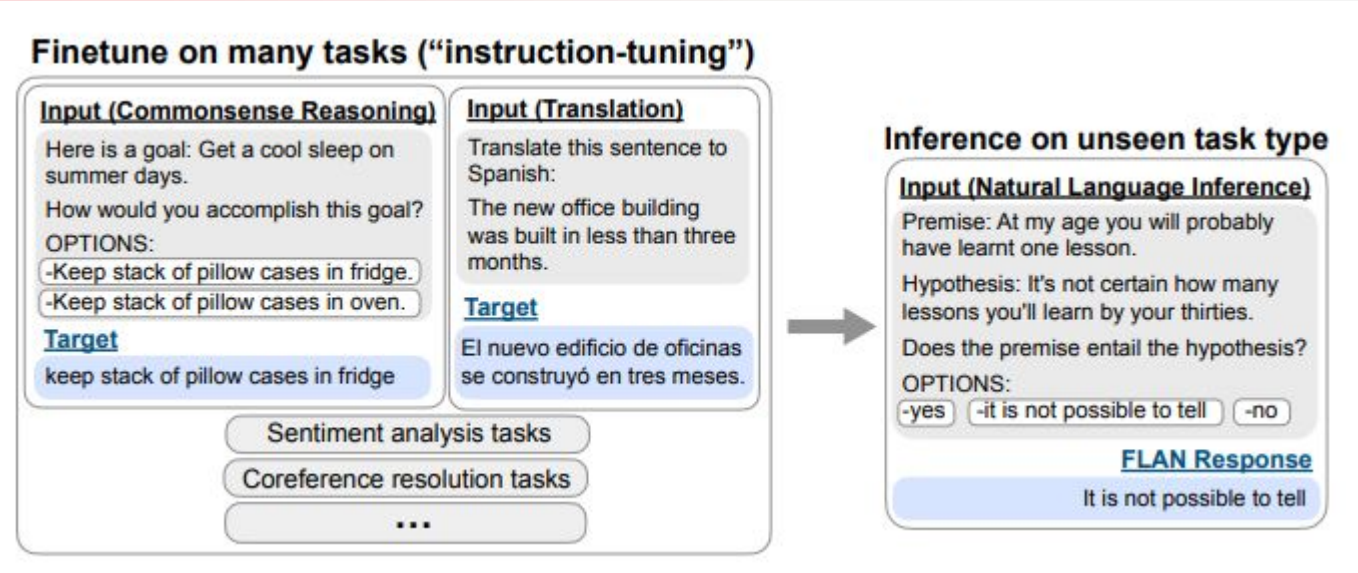
Higher probability output is chosen

Generation output (free-text)

Natural output for FLAN because it is decoder-only and respond in free-text

OPTIONS

- Key to the classification as it gives the model a finite number of outputs that we are looking for
  - Allows the model to be aware of the choices desired



FLAN: Models Used + Training

- The paper primarily used LaMBDA-PT as the model for instruction tuning
  - Decoder-only transformer LM
    - Generates token -> key to OPTIONS
  - 137B parameters (large!)
  - Used 2.49T tokens in pretraining
    - Including web documents, dialog data, and wikipedia
    - Mostly English (90-10)
- Each dataset capped at 30k examples

Results - Models

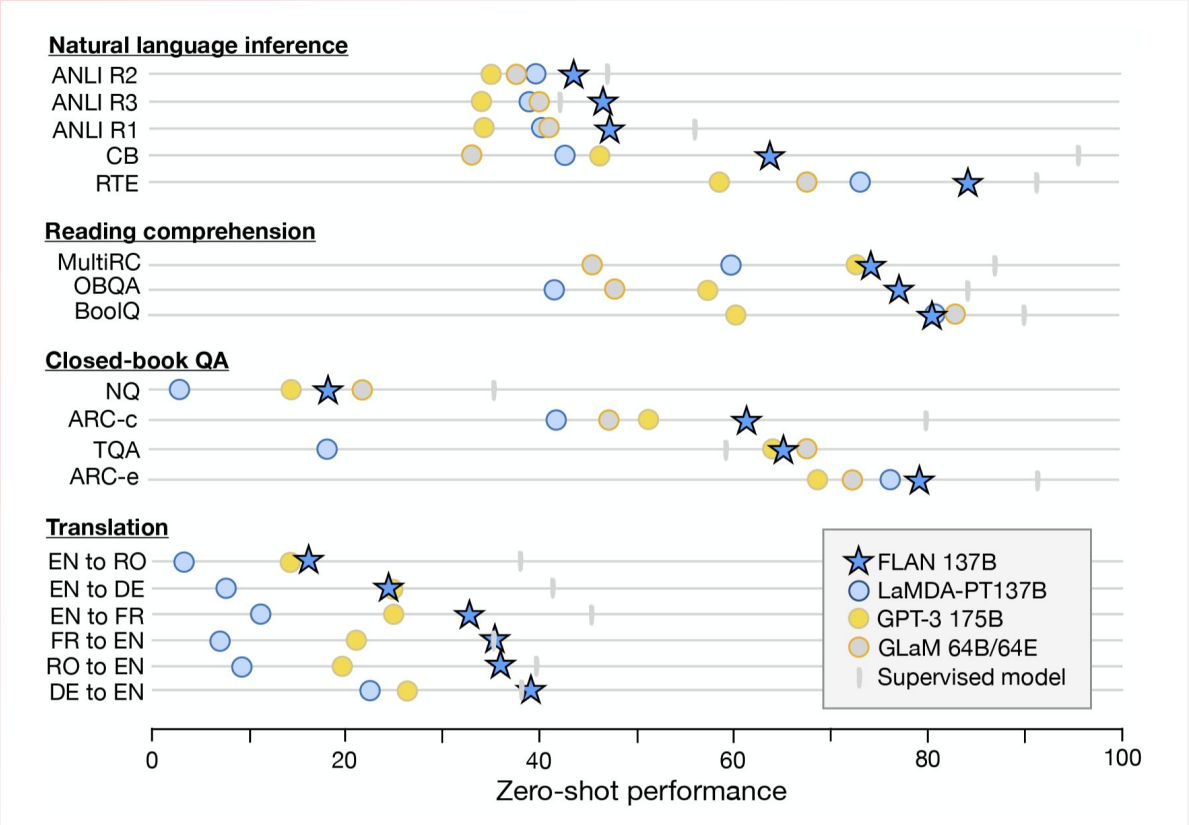
- Overall, the results of zero-shot FLAN are tested against:
  - Base model
  - GPT-3 zero-shot
  - GPT-3 few-shot
- Outperforms zero-shot GPT-3 on 20/25 datasets
- Surpasses GPT-3's few-shot performance on 10/25 datasets
- Outperforms zero-shot GLaM on 13/19 available datasets and one-shot GLaM on 11/19 datasets.

Results - Instruction Type (cluster)

- Overall, we observe that instruction tuning is very effective on tasks **naturally verbalized** as instructions

- Less effective on tasks **directly formulated as language modeling**, where instructions are largely redundant

Results - Instruction Type (cluster)



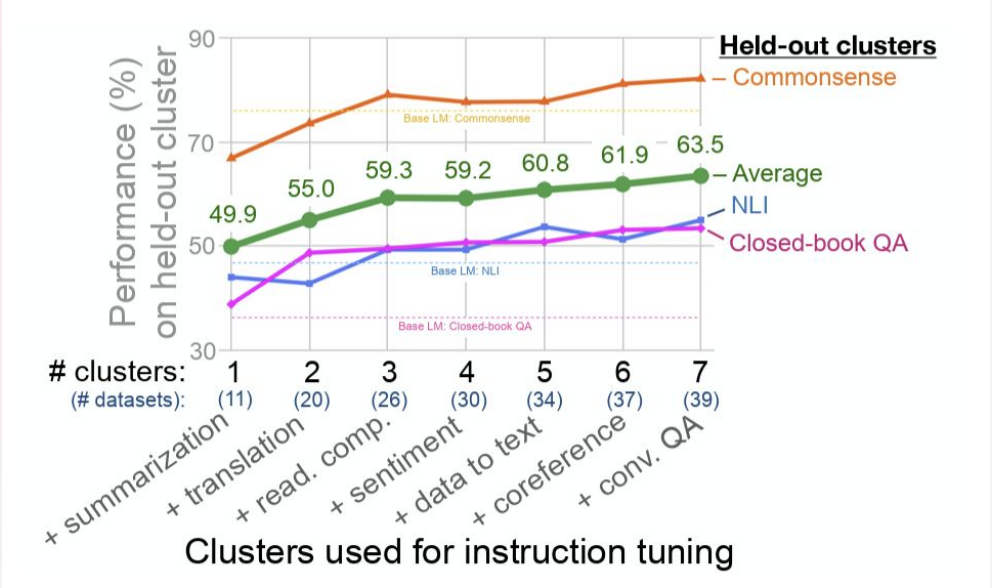
*Supervised models are T5/BERT

When Does Instruction Tuning Help?

- Instruction tuning improves instruction interpretation
- Instruction tuning is highly effective for:
  - NLI (natural language inference)
  - Q and A
  - Translation
  - Struct-to-text
- Not as much performance gains for
  - Commonsense reasoning (sentence completion)
  - Coreference resolution (fill-in-the-blank)

Increasing Performance

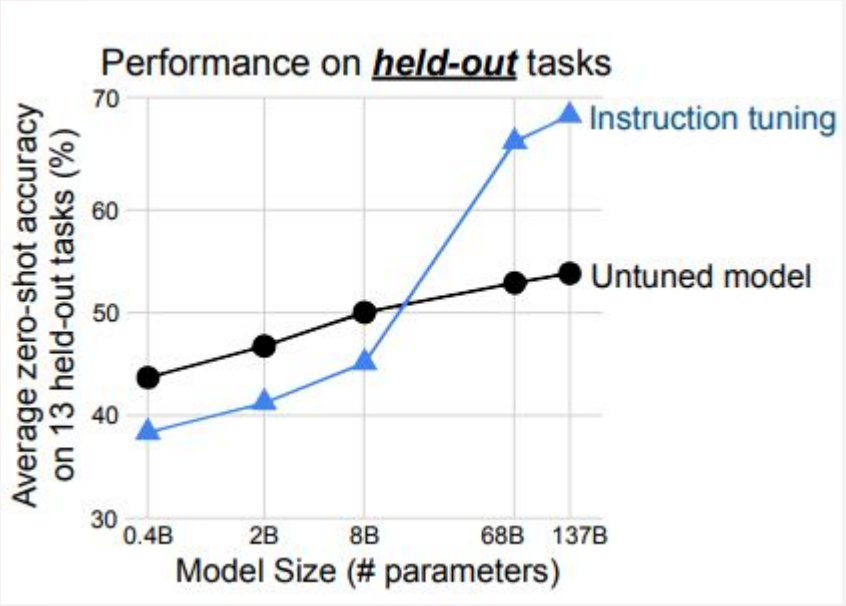
- Performance increases as clusters and datasets are added
- Only 7 clusters added over time here, but more could lead to increased performance



Small vs Large Models

- Small models hurt by instruction tuning
  - Not enough capacity to internalize cross-task patterns

- Large models strengthened
  - Can decipher meta-patterns between tasks



Zero-Shot Performance Gains

- Different methods tested for fine-tuning (**FT**) and evaluation methods (**eval**)

No instruction - instruction
Dataset name - instruction
Dataset name - dataset name
Instruction - instruction (FLAN)

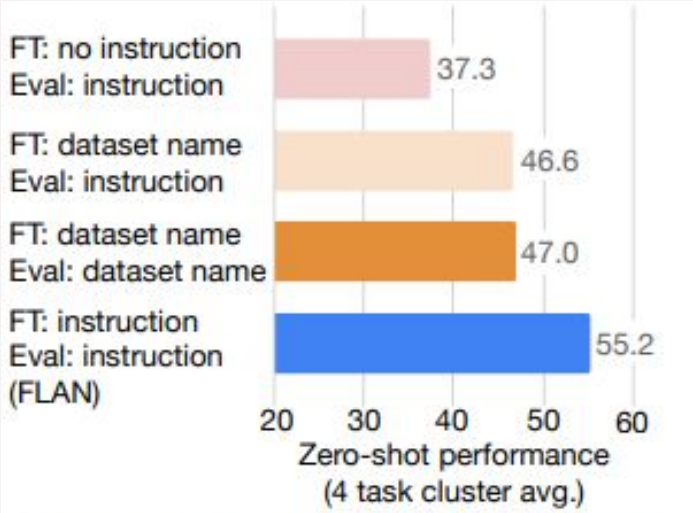


Figure 8: Ablation study result using models with instructions removed from finetuning (FT).

- Instruction wording teaches the model how to interpret tasks, not just memorize patterns

Would Few-Shot Further Improve Performance?

- Instruction tune first, then use few-shot to try to improve performance
- Large/complex output spaces: struct to text, translation, and closed-book QA (exemplars help the model better understand the output format)

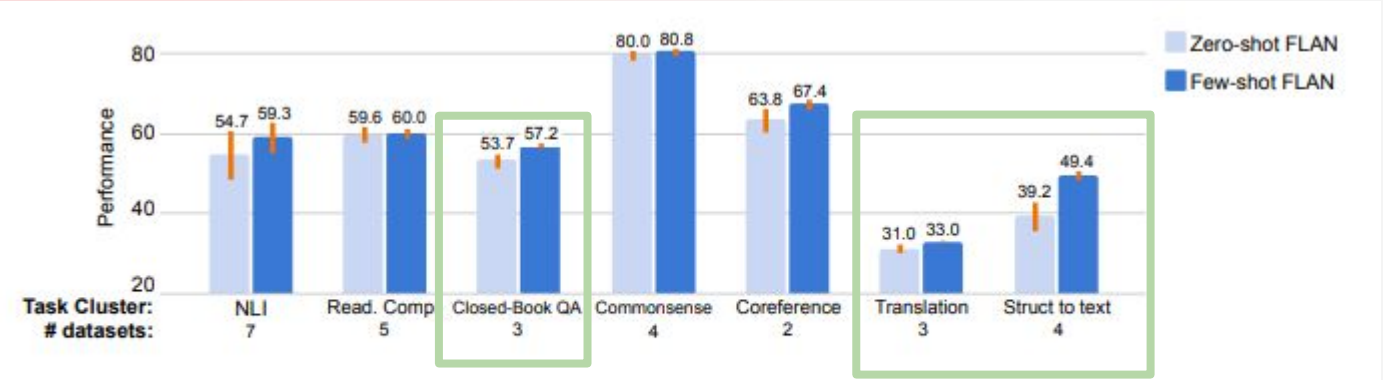


Figure 9: Adding few-shot exemplars to FLAN is a complementary method for improving the performance of instruction-tuned models. The orange bars indicate standard deviation among templates, averaged at the dataset level for each task cluster.

Checkpoint for Prompt Tuning

- Two models: FLAN + LaMBDA-PT
- We use the same instruction tuning on FLAN
- Combine:
  - Instruction tuning (weight updates)
  - Prompt tuning (soft prompts)
- “Freeze” the model
- Instruction tuning produces a better, more adaptable **checkpoint** for downstream NLP tasks

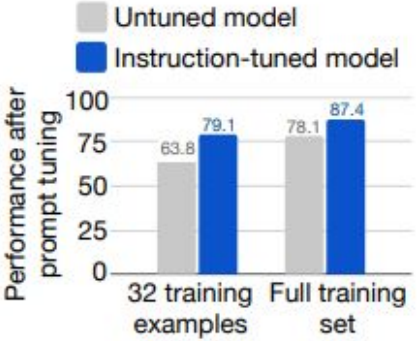


Figure 10: Instruction-tuned models respond better to continuous inputs from prompt tuning. When prompt tuning on a given dataset, no tasks from the same cluster as that dataset were seen during instruction tuning. Performance shown is the average on the SuperGLUE dev set.

Takeaways

1. # of task clusters is key in instruction tuning
2. Scale and diversity lead to results
3. Instruction tuning can help with non zero-shot NLP tasks as well

- Instruction tuning is most helpful when:
  - The task requires mapping a natural language instruction to a **specific structured output**
- It is less helpful when:
  - The task is already just next-token prediction.

Limitations

- This zero-shot instructions tuning requires very large models
- Needs supervised datasets
- Doesn’t help tasks already close to objective
- Still highly dependent quality of instruction datasets
- Instruction training format (templates) = very simple
    - In real applications, the instruction format may be more complex

Cross-Task Generalization via Natural Language Crowdsourcing Instructions

ACL 2022

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Hannaneh Hajishirzi

Motivation: Why Cross-Task Generalization?

The Problem

- Traditional NLP models are task-specific
- A QA model can't do classification
- Fine-tuning needed for every new task
- No labeled data = no model

Human Ability

- Humans read instructions and solve new tasks
- Crowdworkers do this every day
- No task-specific training needed
- A few examples suffice

Can we build models that generalize to new tasks by reading instructions?

Why This Paper Matters

Changed the Paradigm

Before this paper, cross-task generalization was under-explored. Models were trained and tested on the same task. This work showed that natural language instructions can be the bridge to unseen tasks — an idea now central to ChatGPT, Claude, and modern LLMs.

Revealed Surprising Facts

Small fine-tuned models (140M) can beat models 1000x larger (GPT-3 175B) at following instructions. Negative examples actually hurt rather than help. Different task types need different instruction elements.

Created a Reusable Benchmark

The NATURAL INSTRUCTIONS dataset and unified schema became a foundation for the field. It was later scaled to 1600+ tasks (Super-NatInst) and directly inspired instruction-tuning at scale.

This is one of the foundational papers of the instruction-tuning movement

The Core Idea

TRAINING

Seen tasks with instructions
+ input/output pairs

LEARN

EVALUATION

Unseen tasks: only instructions
No task-specific labeled data

Example:

Grammar Check | **QA** | **Tagging** | **Question Typing**

Seen Tasks (Training)

Unseen (Eval)

Instance-Level vs. Task-Level Generalization

Instance-Level (Traditional)

Training: *X_train, Y_train*

Evaluation: $x \rightarrow y$
where (x, y) ∈ (X_test, Y_test)

Same task for train and test

No instructions used

Task-Level (This Paper)

Training: *(I_t, X_t, Y_t) for t ∈ T_seen*

Evaluation: $(x, I\_t) \rightarrow y$
where t ∈ T_unseen

T_unseen ∩ T_seen = ∅

Instructions enable generalization!

Key insight: Instructions (I_t) serve as the bridge between seen and unseen tasks

The NATURAL INSTRUCTIONS Dataset

61	193k	9	6
Distinct Tasks	Task Instances	Source Datasets	Task Categories

Source Datasets:

CosmosQA	DROP	Essential Terms
MC-TACO	MultiRC	QASC
Quoref	ROPES	Winogrande

The Instruction Schema

Title	High-level task description & skill
Definition	Core detailed instructions for the task
Things to Avoid	Undesirable annotations to steer away from
Emphasis/Caution	Key highlighted statements
Positive Examples	Input/output + reasoning for correct behavior
Negative Examples	Input/output + reasoning + suggestions to fix
Prompt	Short command connecting instructions to input

This unified schema enables systematic study of which instruction elements matter most.

Example: MC-TACO Question Generation

Title: Writing questions on "event duration"

Definition: Write a question about event duration based on a given sentence. Event duration = how long events typically last.

Things to Avoid: Don't create questions with explicit answers in text.

Positive Example:

Input: Jack played basketball after school, very tired.
Output: How long did Jack play basketball?
Reason: Asks about event duration → valid.

Negative Example:

Input: He spent two hours on his homework.
Output: How long did he do his homework?
Reason: Answer directly in text → NOT what we want.

Experimental Setup

Task Splits

Random Split
12 eval tasks, 49 training tasks (2 per category)

Leave-one-category
Hold out entire task category

Leave-one-dataset
Hold out all tasks from one source dataset

Leave-one-task
Hold out a single task

Models

BART (base)

140M params • Encoder-decoder
Fine-tuned on T_seen tasks
Trained 3 epochs, LR = 5e-5

GPT-3

175B params • Autoregressive
No fine-tuning • Zero-shot only
davinci-instruct engine

Metric: ROUGE-L (text generation evaluation)

How Instructions Are Encoded

No Instruction	input: x ⟶ output:
Prompt	Prompt + input
Prompt + Definition	Definition + Prompt + input
Prompt + Things to Avoid	Avoid + Prompt + input
Prompt + Pos. Examples	Examples + Prompt + input
Full Instruction	All fields + examples + input

Purpose: Ablate which elements of instructions contribute to generalization — this design enables the paper's key finding that different elements serve different task types.

Key Result: Instructions Help Generalization

model ↓	evaluation set $\mathcal{T}_{\text{unseen}}$ →	random split of tasks	leave-one-category (QG)	leave-one-dataset (QASC)	leave-one-task (QASC QG)
BART (fine-Tuned)	NO INSTRUCTIONS	13	6	37	20
	FULL INSTRUCTIONS	**32**	**17**	**51**	**56**
GPT3 (not fine-tuned)	FULL INSTRUCTIONS	24	33	22	33

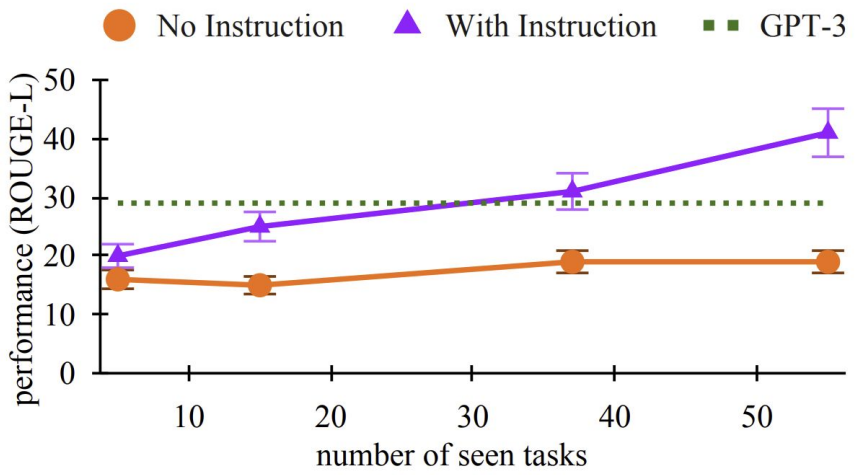
Table 4: Cross-task generalization of BART under various splits (§5.1). Fine-tuned BART shows improved performance when provided with instructions. It also archives better performance than GPT3, despite being over $1k$ times smaller. All numbers are ROUGE-L.

Which Instruction Elements Matter?

model ↓	task category →	QG	AG	CF	IAG	MM	VF	avg
BART (fine-tuned)	NO INSTRUCTION	26	6	0	21	33	7	13
	PROMPT	27	22	7	22	34	**9**	20
	+DEFINITION	35	24	50	25	36	7	30↑ (+50)
	+THINGS TO AVOID	33	24	4	24	**58**	**9**	25↑ (+25)
	+EMPHASIS	38	23	16	**26**	49	3	26↑ (+30)
	+POS. EXAMPLES	53	22	14	25	17	7	23↑ (+15)
	+DEFINITION+POS. EXAMPLES	51	23	**56**	25	37	6	33↑ (+65)
	POS. EXAMP.	**55**	6	18	25	8	6	20
	FULL INSTRUCTION	46	**25**	52	25	35	7	32↑ (+60)
GPT3 (not fine-tuned)	FULL INSTRUCTION	33	18	8	12	60	11	24 (+11)

Table 5: Cross-task generalization under random split (§5.1). Models show improved results when provided with instructions. The numbers in parenthesis indicate absolute gains compared to 'NO INSTRUCTIONS' baseline. Fine-tuned BART archives better performance than GPT3, despite being over $1k$ times smaller. Category names: QG: Question Generation, AG: Answer Generation, CF: Classification, IAG: Incorrect Answer Generation, MM: Minimal Text Modification, VF: Verification. All numbers are ROUGE-L (in percentage).

Scaling: More Seen Tasks → Better Generalization



(b) BART evaluation on *unseen* tasks (y-axis is perf. on $\mathcal{T}_{\text{unseen}}$) when supervised with *seen* tasks (x-axis is $|\mathcal{T}_{\text{seen}}|$). A model using instructions (I_t) consistently improves with more observed tasks. In contrast, models with no access to the instructions show no sign of improved generalization. Details in §6.3.

Key Findings

- **With instructions: consistent upward trend**
- Without instructions: flat — no benefit from more tasks
- Suggests further scaling could yield even better models

Surprising Findings

Negative Examples Hurt Performance

Full instructions were tested with and without the negative example fields. Everything else stayed the same. Removing them improved scores. (Random split, ROUGE-L %)

	Full Instr. (with neg. ex.)	Full Instr. (neg. ex. removed)	Change
BART	32%	35%	+3 pp
GPT-3	24%	44%	+20 pp

Models are confused by being told what NOT to do.
GPT-3 is especially sensitive — 20 pp worse with neg. examples.

Why? Possibly because negative examples look similar to positive ones in the input — the model may learn the wrong pattern from them.

How Far From Supervised Learning?

To measure the gap, the authors trained a separate BART model on each eval task's own labeled data (the traditional approach) and compared to the cross-task model.

66%

Trained on each task's own data
12 separate BART models, one per eval task
= Standard supervised learning

32%

Trained on 49 other tasks, never seen eval tasks
Only reads instructions at test time
= Cross-task generalization (this paper)

34 pp gap → promising, but far from solved

Limitations & Critiques

Limited Scale & Task Diversity	Only 61 tasks from 9 QA-focused datasets. All textual, no multimodal tasks. Heavy QA bias may not generalize to other NLP domains (summarization, translation, dialogue).	*Addressed by Super-NatInst (1600+ tasks), but the original results may overfit to QA-adjacent tasks.*
ROUGE-L as Sole Metric	ROUGE-L measures surface text overlap, not semantic correctness. A grammatically different but correct answer may score poorly. No human evaluation on model outputs at scale.	*Future work could use BERTScore, human evaluation, or task-specific metrics (accuracy for classification).*
BART Base Only (140M)	Only tested on one fine-tuned model size. No scaling analysis across model sizes (e.g., BART-large, T5-XL). Unclear if conclusions hold for larger encoder-decoder models.	*Later work (FLAN, T0) confirmed benefits at larger scale, but this paper doesn't show it.*
Schema Mapping is Manual & Subjective	Mapping raw instructions to schema took 6-34 hrs/task with subjective editorial decisions. Difficult to scale or reproduce. Different annotators might create different instructions.	*Could explore automated instruction extraction or standardized annotation protocols.*

Despite these limitations, the core insights — that instructions help and that different elements serve different tasks — have been widely validated.

SUPER-NATURALINSTRUCTIONS: Generalization via Declarative Instructions on 1600+ NLP Tasks

EMNLP 2022

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, Daniel Khashabi

What makes it difficult to evaluate the performance of a model at instruction-following?

- Much harder to obtain training data
- Difficult to create a metric that approximates human evaluation
- Hard to identify true generalization in a model trained on such a massive amount of data

The key question:

Do models actually generalize to unseen instructions, or do they interpolate within seen task formats?

Answering this question requires understanding:

- How to measure generalization
- How to construct training data that promotes it
- What architectural or training choices matter

What should a generalization dataset look like?

- Task instructions (most important)
- Negative example
- Positive example
- Problem instance

In a traditional NLP dataset, we would only need the last two!

Task Type	Word Analogy
Task ID	task1156_bard_word_analogy
Definition	Two analogies that relate actions to the tools used to perform the action is given in the form "A : B. C : ?". "A : B" relates action A to tool B. Your task is to replace the question mark (?) with the appropriate tool for the given action C, following the "A : B" relation.
Positive Example	**Input**: eat : fork. cook : ? **Output**: pan **Explanation**: The given analogy relates actions to the tools used to perform them. A fork can be used to eat. To cook, a pan can be used.
Negative Example	**Input**: dig : shovel. wash : ? **Output**: sink **Explanation**: The given analogy relates actions to the tools used to perform them. A knife can be used to cut. To wash, a sink CANNOT be used.
Instance	**Input**: cut : knife. wash : ? **Valid Output**: ["soap", "washcloth", "detergent", "rag"]

What is difficult about building a dataset like this?

How to create massive datasets without massive human effort?

Key Idea:

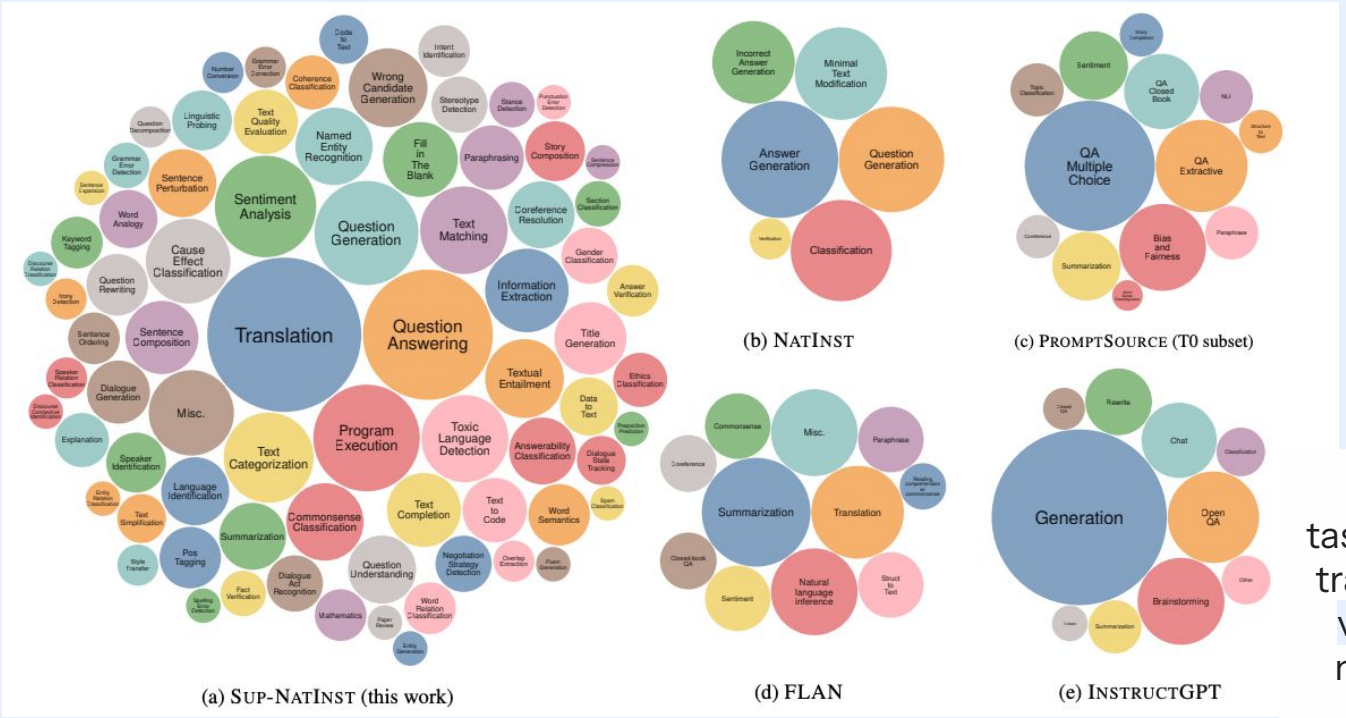
Have human curators generate a locus of seed tasks, then use another LLM to create variations

This means no new human labeling is required at scale, although the generated tasks may be related in subtle ways (more on this later)

Pipeline

1. **Seed tasks**
   - A small, manually curated set of diverse instructions.
   - Covers classification, transformation, reasoning, generation, etc.
2. **Instruction generation**
   - A pretrained LLM is prompted to generate:
     - New instructions
     - Input examples
       Corresponding outputs
3. **Filtering & deduplication**
   - Remove low-quality or duplicate instructions.
   - Ensure diversity across task types.
4. **Bootstrapping loop**
   - Generated data becomes new conditioning context.
   - Iteratively expands task space.

Result



(a) SUP-NATINST (this work) (b) NATINST (c) PROMPTSOURCE (T0 subset)

(d) FLAN (e) INSTRUCTGPT

The prevailing task-following models were trained on a much smaller variety of tasks. Can we make something better using Sup-NatInst?

Result

Resource →	SUP-NATINST (this work)	NATINST (Mishra et al., 2022b)	CROSSFIT (Ye et al., 2021)	PROMPTSOURCE (Bach et al., 2022)	FLAN (Wei et al., 2022)	INSTRUCTGPT (Ouyang et al., 2022)
Has task instructions?	✓	✓	✗	✓	✓	✓
Has negative examples?	✓	✓	✗	✗	✗	✗
Has non-English tasks?	✓	✗	✗	✗	✓	✓
Is public?	✓	✓	✓	✓	✓	✗
Number of tasks	1616	61	269	176	62	–
Number of instructions	1616	61	–	2052	620	14378
Number of annotated tasks types	76	6	13	13*	12	10
Avg. task definition length (words)	56.6	134.4	–	24.8	8.2	–

Contemporary models

T5

- General LLM by Google Research
- Trained only on web corpus

T0

- One of the earliest LLMs designed for zero-shot tasks
- Fine-tuned from base T5 model
- Trained on P3 dataset reformatted as tasks

InstructGPT

- Based on GPT-3
- Supervised fine-tuning (SFT) on human-written instruction–response pairs
- Reward model trained on human preference rankings
- Reinforcement learning from human feedback (RLHF)
- Best results before this paper, but likely expensive to train and aligned to user intent over generalization

T*k*-Instruct: Architecture

Base Model: Google Research T5 Family

Tk-Instruct builds on:

- Same T5 base model that T0 used
- Specifically large-scale encoder–decoder variants

Encoder–Decoder Structure

- Encoder processes:
  - Task definition
  - Positive examples
    Negative examples (optional)
  - Input instance
- Decoder generates:
  - Output text conditioned on encoded representation

Notably, T*k*-Instruct was about 16x smaller than previous successful task-learning models, requiring just 11 billion parameters compared to ~175 billion used by InstructGPT

T*k*-Instruct: Training

- Tasks are sampled uniformly (or rebalanced).
- Avoid dominance of high-resource tasks.
- Encourage equal exposure to heterogeneous task types.

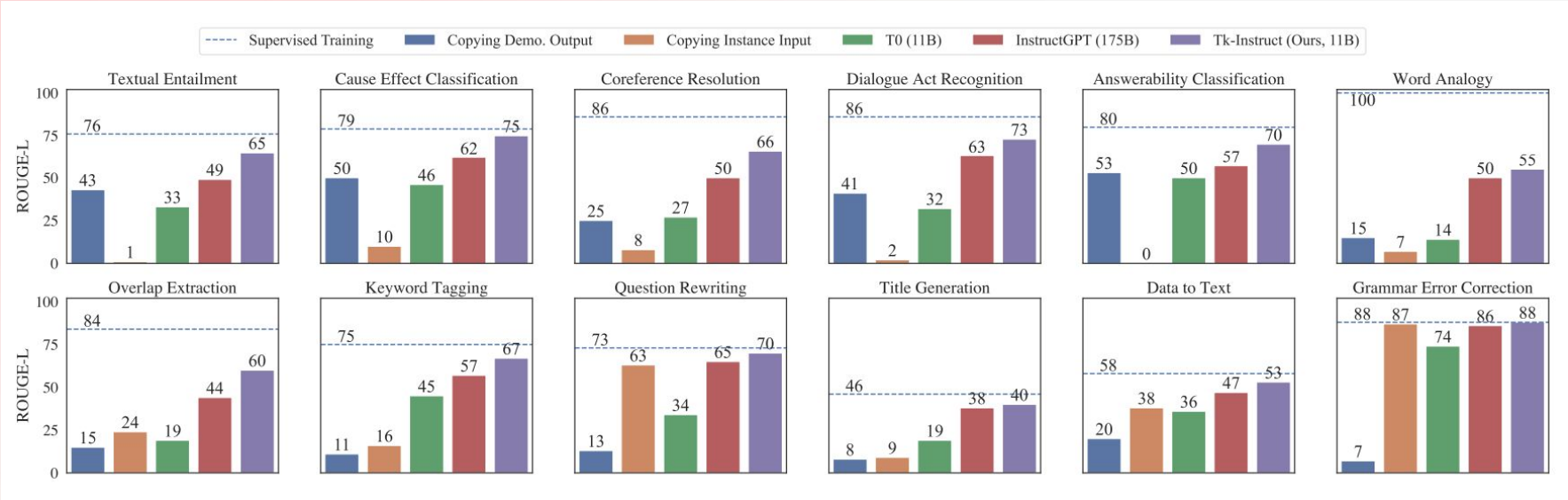
Key Training Mechanisms

1. **Held-out task splits**
   * Entire tasks removed during training.
     - Evaluated zero-shot.
2. **Instruction diversity exposure**
   - Varying phrasings.
   - Multiple example configurations.
3. **Instance-level shuffling**
   - Prevents overfitting to fixed formats.

Although the dataset is one of the first to include negative examples, they were not used to train the model!

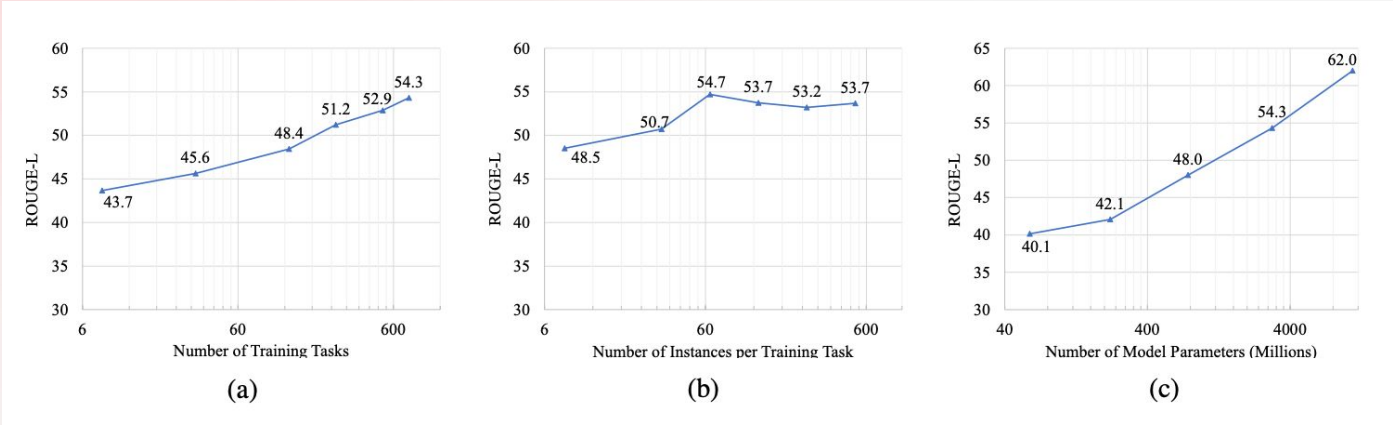
Evaluation was divided into two tracks, English and Cross-Language. The model performed similarly on both tracks.

T*k*-Instruct: Performance



T*k*-Instruct: Takeaways

1. Instruction-tuning clearly enables stronger generalization to unseen tasks
2. Larger, more varied training datasets allow performance to scale more quickly
    a. Model generalization performance grows log-linearly as the set of tasks used for training increases
3. A large number of training instances do not help generalization
    a. the model's performance saturates when only 64 instances per task are used for training



Benefit of different instruction elements

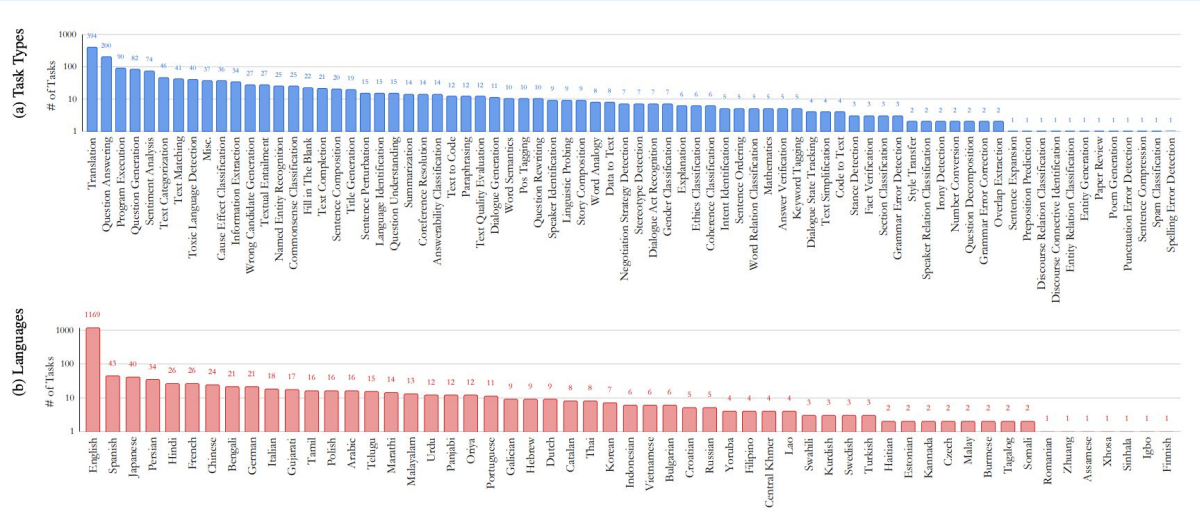
Testing Encoding → / Training Encoding ↓	Task ID	Def	Pos (1)	Def + Pos (1)	Pos (2)	Def + Pos (2)	Def + Pos (2) + Neg (2)	Def + Pos (2) + Neg (2) + Expl	Pos (4)	Def + Pos (4)	Average
Task ID	21.2	33.3	16.8	30.9	23.0	33.7	33.9	31.6	26.0	36.4	33.9
Def	17.3	45.0	31.1	43.8	36.4	46.4	44.2	44.3	38.0	46.0	39.9
Pos (1)	10.9	22.1	43.9	47.8	46.6	49.2	46.2	43.4	46.6	49.5	43.1
Def + Pos (1)	11.1	42.2	43.8	52.4	47.4	53.3	53.1	51.8	47.8	53.7	44.5
Pos (2)	12.7	22.4	47.1	50.2	49.3	52.3	50.6	46.7	49.8	52.4	45.0
Def + Pos (2)	12.4	42.1	44.5	52.4	49.0	54.3	53.5	52.7	50.3	54.8	46.4
Def + Pos (2) + Neg (2)	14.0	42.3	43.6	51.8	48.6	53.5	54.3	50.2	49.6	53.8	45.9
Def + Pos (2) + Neg (2) + Expl	15.4	42.0	43.8	50.7	47.6	51.9	52.5	52.6	48.6	52.2	44.3
Pos (4)	11.0	23.9	45.6	49.8	49.0	51.7	49.5	47.5	49.8	51.3	44.5
Definition + Pos (4)	11.0	42.4	44.3	51.9	48.7	53.7	53.4	50.6	50.5	53.5	46.0

Notice that definition-only models don't generalize to example-only test encodings and vice versa. However, models trained on encodings that contain both definition and examples perform well across the board

- Including the task definition consistently helps the model to generalize better
- Combining the task definition with positive demonstration examples yields further improvement
- However, adding more demonstration examples is negligible
- Negative examples help a little bit (unlike previous paper); explanations decrease performance

Limitations

- This dataset suffers from significant skew issues in both task and language variety
  - How effectively can this be countered with uniform sampling?
- Also, the possibility that using an LLM to generate tasks introduces bias is largely unaddressed
- ROUGE-L evaluation metric is not perfect, although finding a better alternative is difficult



Implications for future model design

1. Data > architecture (for this problem class)
2. Instruction representation is central
3. Generalization depends on:
    - Task manifold coverage
    - Semantic variety
    - Reformulation exposure

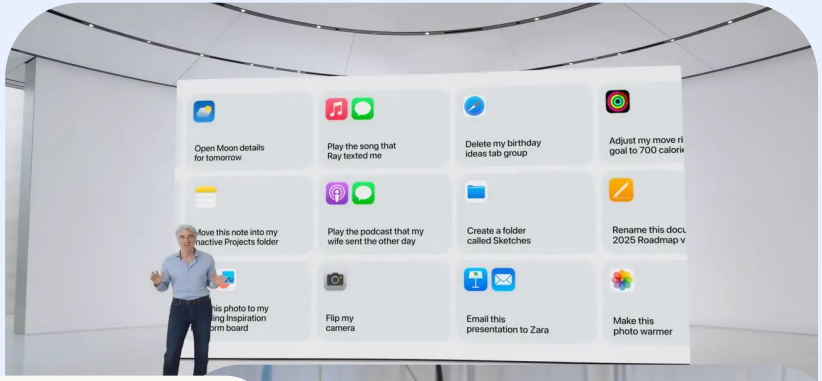
Future directions:

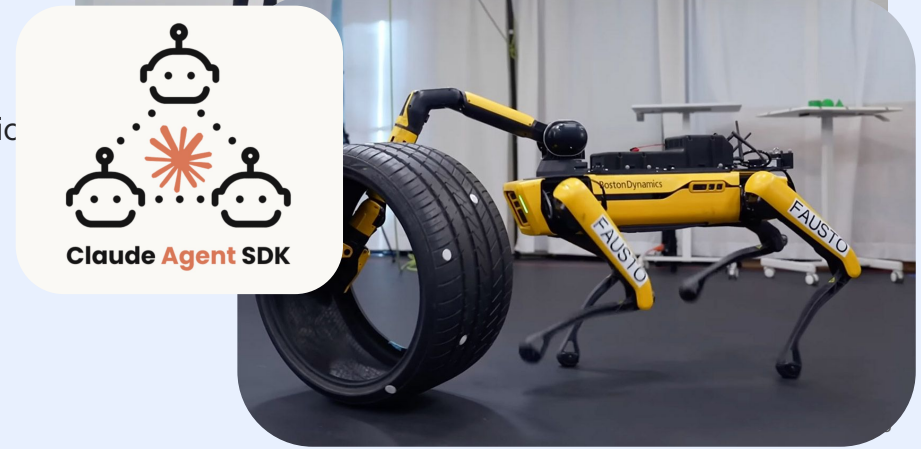
- Explicit task embeddings
- Self-instruction
- Compositional instruction training
- Adversarial instruction robustness testing

Variety of data is more important than quantity

Alternative use cases

- All three papers discussed today still focus entirely on language tasks
- Instruction tuning could be a precursor to agentic AI
  - Apple Intelligence, Microsoft Copilot, etc.
- If models can be trained to interface with larger systems, they can be capable of things like:
  - Interacting with an OS
  - Translating human instructions into physical actic
  - Navigating through human spaces





In conclusion

LLMs can be trained to follow new, specific instructions

This can be done through prompting a large base model, fine-tuning a base model on instruction data, or designing a model specific to multi-task operation.

The format, distribution, and variety of these instructions matters

Including too much description, not enough task variety, or formatting the input incorrectly can make it more difficult for a model to generalize.

Performance scales strongly with both better data and larger models

Using high-quality datasets with a large variety of instructions, and training models from the ground up using them, enables strong, efficient zero-shot performance.

Thank you

More about Sup-NatInst

statistic	
# of tasks	1616
# of task types	76
# of languages	55
# of domains	33
# of non-English tasks	576
avg. definition length (words per task)	56.6
avg. # of positive examples (per task)	2.8
avg. # of negative examples (per task)	2.4
avg. # of instances (per task)	3106.0

Table 2: Statistics of SUP-NATINST.

"Contributors were encouraged to be creative and source the tasks from several resources: (a) existing public NLP datasets, (b) available intermediate annotations in crowdsourcing experiments (e.g., paraphrasing questions or rating their quality during crowdsourcing a QA dataset), or (c) synthetic tasks that can be communicated to an average human in a few sentences (e.g., basic algebraic operations like number comparison, finding the longest palindrome substring, etc.)."