

Scalable Data Paradigms for Steering General-Purpose Language Models

Yizhong Wang

A dissertation

submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2025

Reading Committee:

Hannaneh Hajishirzi, Co-Chair

Noah A. Smith, Co-Chair

Pang Wei Koh

Program Authorized to Offer Degree:

Computer Science and Engineering

© Copyright 2025

Yizhong Wang

University of Washington

Abstract

Scalable Data Paradigms
for Steering General-Purpose Language Models

Yizhong Wang

Co-chairs of the Supervisory Committee:

Hannaneh Hajishirzi

Department of Computer Science and Engineering

Noah A. Smith

Department of Computer Science and Engineering

Pretrained Language Models (LMs) have demonstrated remarkable general-purpose capabilities by encoding vast amounts of knowledge from the internet. However, effectively steering these models to serve diverse downstream applications, such as following instructions, chatting with users, using tools, or performing complex reasoning, poses another set of challenges that require diverse, high-quality, and increasingly costly training data.

This dissertation explores scalable paradigms for structuring, creating, and optimizing data to facilitate the broader generalization of language models and enhance their critical capabilities. First, through the creation of the SUPER-NATURALINSTRUCTIONS benchmark—a large-scale dataset with over 1,600 NLP tasks—I demonstrate that unifying NLP tasks via natural language instructions enables model generalization at the task level. Second, I propose SELF-INSTRUCT, a novel framework where LMs generate their own instructional data to train themselves, thereby demonstrating model self-improvement. Third, I develop HYPER, a framework that routes preference annotation tasks between humans and AI to optimize data

quality and collection efficiency for preference-based learning. Finally, I systematically study the impact of diverse open instruction-tuning datasets on LM capabilities, leading to the development of the TULU series of openly available and highly capable models.

Together, these efforts—unifying task structures, leveraging model-generated synthetic data, optimizing human-AI data partnerships, and fostering open data ecosystems—have demonstrated an effective path to building a strong, scalable, and community-driven data foundation for post-training language models. Finally, I envision future directions that can further enhance this data foundation for building more advanced and sustainable AI systems.

Acknowledgements

Six years ago, when I set out on this PhD journey, I could never have imagined how much the world—and my own life—would change. The sudden outbreak of the COVID-19 pandemic in my first year, the rise of ChatGPT, and the recent tightening of research funding have all reshaped the scientific landscape around me. At the same time, I stepped onto the fast-moving track of large language models, trained models that even surprised me, celebrated my marriage, mourned the loss of my grandfather, welcomed my first child, and realized my dream of a faculty career. These intertwined moments of challenge and joy have become my singular, priceless experience that I will carry for the rest of my life. Through it all, I owe my deepest thanks to the people who stood by me at every step.

First and foremost, I would like to thank my advisors, Hannaneh Hajishirzi and Noah Smith. From our very first meeting, they encouraged me to think boldly and pursue research that could truly matter, and then provided the freedom, guidance, and resources to make it happen. Their belief in me never wavered, even during my first two years, when I was struggling to publish my first paper and doubting myself. Looking back now, I realize that the ideas we conceived together—test-time training and intermediate reasoning steps—could have been great successes, but we were ahead of our time, before pretraining autoregressive language models actually worked. I remember Noah trying to lift the weight off my shoulders by telling me that “research is a marathon, not a sprint.” In addition, Hanna mapped my half-formed interests onto ongoing projects at AI2, opening doors to collaborations that finally led to success. I feel doubly fortunate to have benefited from their complementary styles: Hanna walked me through concrete brainstorming, project execution details, and how to nudge work toward real-world impact, while Noah taught me how to think about research itself—the rigorous attitude, the meta skills for doing things well, and a long-view understanding of where the field comes from and where it is headed. They will forever be role models for my academic career.

My sincere thanks also go to my committee members, Pang Wei Koh and Shane Steinert-Threlkeld, for their valuable feedback on framing and improving this dissertation. Pang Wei—together with the rest of the UWNLP faculty, Luke Zettlemoyer, Yejin Choi, and Yulia Tsvetkov—has been a steady source of guidance throughout my PhD. They also provided tremendous support and patient advice during my job search. I feel honored to have co-authored at least one paper with each of them; their generosity and collaborative spirit make this community so special. I also want to thank other faculty members and staff at UW CSE, especially Magdalena Balazinska, Shyam Gollakota, Sheng Wang, Ali Farhadi, and Elise DeGoede, for their kind support during my PhD.

I have been very fortunate to grow within the lively UWNLP community. Senior students and postdocs set examples I still aspire to: Swabha Swayamdipta mentored me early in my PhD on our dataset-cartography project and opened my eyes to data-centric ML; Tao Yu, a close friend since my second year, exchanged spicy thoughts with me on many trending topics; and Sewon Min offered steady help not only in research but also in life—and, most recently, in the job search and career decisions. I was lucky to sit in CSE 318, which gave me the best research buddies. Akari Asai, Jungo Kasai, Sewon Min, Tim Dettmers, and Victor Zhong were the greatest collaborators and the best break-time companions. I still miss the group activities Victor organized and the evenings when Tim arrived with his handmade tiramisu. Today, Stella Li, Jacqueline He, and Rui Xin have taken over the space and brought new fun. Beyond the office walls, many collaborators have become friends I treasure: Alisa Liu, Bowen Zhao, Gabriel Ilharco, Hamish Ivison, Hao Peng, Jiacheng Liu, Joel Jang, Qingqing Cao, Rahul Nadkarni, Vidhisha Balachandran, Weijia Shi, Xiaochuang Han, Yushi Hu, Zeqiu (Ellen) Wu, Zeyu (Leo) Liu, and Zhiyuan Zeng, together with the wider H2Lab and ARK Lab communities. Many of my ideas in this dissertation were inspired by them during our daily conversations and group activities.

The Allen Institute for AI (AI2) has been a significant part of my PhD journey. I was lucky to intern there multiple times and to spend my last two years as a student researcher. Three mentors have guided this journey with more patience and generosity than I can measure: Matt Gardner flew me to the AI2 Irvine office for my first internship and introduced me to the UW community; Daniel Khashabi supervised two of my most influential projects, patiently teaching me foundational research skills; and Pradeep Dasigi closely discussed and supported me on the TULU projects. Along the way, an extraordinary group of AI2 colleagues also kept

the energy high and the ideas flowing: Ashish Sabharwal, David Wadden, Dirk Groeneveld, Faeze Brahman, Hamish Ivison, Iz Beltagy, Jacob Morrison, Jack Hessel, Khyathi Raghavi Chandu, Kyle Lo, Lester James V. Miranda, Luca Soldaini, Matthew Peters, Nathan Lambert, Nouha Dziri, Oyvind Tafjord, Sachin Kumar, Shengyi (Costa) Huang, Tushar Khot, Valentina Pyatkin, Xinxin (Shane) Lyu, Yanai Elazar, and Yuling Gu. I am deeply grateful to have been their colleague and to have learned from all of them.

Outside the lab, my Chinese cohort mates—Dong He, Liwei Jiang, Wei Qiu, Wentao Yuan, Xiangfeng Zhu, Xieyang Xu, and Yuhao Wan—shared the highs and lows that only fellow travelers can truly understand. Beyond Seattle, the wider NLP community has also offered me invaluable support. In particular, I have benefited greatly from Colin Raffel, Eric Wallace, Fangzheng (Frank) Xu, Greg Durrett, Jonathan Berant, Patrick Lewis, Sameer Singh, Sebastian Riedel, Seungone Kim, Sujian Li, Swaroop Mishra, Tatsunori Hashimoto, Vijay Viswanathan, Xiang Yue, Yao Fu, Yeganeh Kordi, and Yuchen (Bill) Lin.

Lastly, none of this would have been possible without my family's unwavering love and sacrifice. My grandparents were my quiet source of strength; even a brief visit or phone call with them could calm the hardest day. Sadly, my grandfather passed away in the summer of 2024, before he could see this milestone, but his gentle pride still guides every step I take. My parents gave me the most loving home I could imagine and shielded me from countless distractions so I could focus on my interests—a lifetime would not be long enough to repay their sacrifices. And to my wife, Haiyan He, I owe more than words can hold. She set aside her own career, left her family, and moved with me to the United States, supporting every turn I made and lifting me whenever experiments failed. Six months ago, she brought our son, Ruiyang (Ryan), into the world, filling my job-search process with pure joy. If this degree were a medal, half of it would belong to her.

DEDICATION

In loving memory of my grandfather,
whose quiet pride and gentle encouragement lit every step of my path.

Contents

1	Introduction	23
1.1	Brief Overview	23
1.2	Background	23
1.3	Challenges	24
1.4	Approach: Scalable Data Paradigms	24
1.5	Dissertation Outline	25
2	Unifying NLP Tasks by Adding Instructions	27
2.1	Overview	27
2.2	Background	29
2.3	SUPER-NATURALINSTRUCTIONS dataset	31
2.3.1	Instruction schema	31
2.3.2	Collection process	31
2.3.3	Diversity of tasks	32
2.4	Teaching Models to Follow Instructions	33
2.4.1	Formulation	33
2.4.2	Tk -INSTRUCT	34
2.5	Results	34
2.5.1	Benchmarking Setup	34
2.5.2	Baselines and Existing Models	35
2.5.3	Main Results	37

2.5.4	Human Evaluation	38
2.6	Further Analysis	39
2.6.1	Scaling Trends of Generalization	39
2.6.2	Instructing with Different Elements	40
2.7	Discussion & Future Work	41
2.8	Summary	42
3	Synthesizing Training Data with Models	43
3.1	Overview	43
3.2	Background	45
3.3	Self-Instruct Pipeline	47
3.3.1	Defining Instruction Data	48
3.3.2	Automatic Instruction Data Generation	48
3.3.3	Finetuning the LM to Follow Instructions	50
3.4	Synthesized Data from GPT3	51
3.4.1	Statistics	51
3.4.2	Diversity	51
3.4.3	Quality	52
3.5	Results	52
3.5.1	GPT3 _{SELF-INST} : a self-instructed GPT3 model	53
3.5.2	Baselines	53
3.5.3	Zero-Shot Generalization on SUPERNI benchmark	54
3.5.4	Generalization to User-oriented Instructions on Novel Tasks	55
3.5.5	Effect of Data Size and Quality	57
3.6	Discussion & Future work	58
3.7	Summary	59
4	Routing Annotations between Human and AI	61
4.1	Overview	61

4.2	Background	63
4.3	Learning to Route Preference Annotations	64
4.3.1	Problem Formulation	64
4.3.2	Performance Prediction Model (PPM)	65
4.3.3	Routing Strategy based on the PPM	68
4.3.4	Routing Strategy for a Single Instance	68
4.4	MULTIPREF: A New Preference Dataset	68
4.5	Experiments	69
4.5.1	Performance Prediction Model Details	70
4.5.2	Generalization to Unseen Datasets	71
4.5.3	Generalization to Other Base Models	73
4.5.4	Generalization to other Evaluation Tasks	73
4.6	Analysis: When are Human Annotations Helpful?	74
4.7	Discussion & Future Work	75
4.8	Summary	77
5	Post-training LMs with Open Data	79
5.1	Overview	79
5.2	Background	81
5.3	Training Models with Various Datasets	82
5.3.1	Unifying the Format	82
5.3.2	Model Training Details	83
5.3.3	TÜLU 🦒: a Better Instruction-Tuned Model by Combining Resources	84
5.4	Evaluation Setup	84
5.4.1	Facets of Evaluation	84
5.4.2	Model-Based Evaluation using GPT-4	86
5.4.3	Human Evaluation	86
5.5	Results	87
5.5.1	Analysis of Instruction Tuning Datasets and Base Models	87

5.5.2	Pushing the Limits of Open Models	89
5.5.3	Evaluation of Potential Risks and Harms	90
5.5.4	Model-Based Evaluation Results for Open-Ended Generation	91
5.5.5	Human Evaluation Results for Open-Ended Generation	92
5.6	Discussion & Future Work	93
5.7	Summary	93
6	Conclusion and Future Work	95
6.1	Conclusion	95
6.2	Future Work	96
6.2.1	Data Optimization and Expansion at Scale	96
6.2.2	Automated Verification for Model Improvement	97
6.2.3	Fair and Responsible Data Practices in AI	97
6.2.4	Data for Real-World Multimodal Interactions	98
A	Appendix: Unifying NLP Tasks by Adding Instructions	121
A.1	Crowdsourcing Human Feedback	121
A.2	Crowdsourcing Human Judgements of Generation Quality	121
A.3	Instruction Schema	123
A.4	Model Implementation Details	123
A.5	Evaluation Metrics	124
A.6	Distribution of Tasks	125
A.7	Evaluation Tasks	127
A.8	Performance Improvement per Evaluation Task	127
B	Appendix: Synthesizing Training Data with Models	139
B.1	Implementation Details	139
B.1.1	Writing the Seed Tasks	139
B.1.2	Querying the GPT3 API	140
B.1.3	Finetuning GPT3	141

B.1.4	Prompting Templates for Data Generation	141
B.2	Human Evaluation Details for Following the User-oriented Instructions	145
B.2.1	Human Evaluation Setup	145
B.2.2	Human Evaluation Agreement	145
B.2.3	Example Predictions from GPT3 _{SELF-INST}	145
B.3	Task and Instance Examples from the Generated Instruction Data	148
C	Appendix: Routing Annotations between Human and AI	151
C.1	Construction of MULTIPREF	151
C.2	Qualification Test for MULTIPREF Annotators	155
C.3	Inference-time Selection Strategies	161
C.4	Complete list of tags	162
C.4.1	Meta-analyzer for descriptive tags	163
C.5	Performance Gain	164
C.6	Best-of-N Evaluation Details	164
C.7	Finegrained RewardBench Results	166
C.8	Direct Preference Optimization Results	166
C.9	Reward Model Training Details	168
C.10	Case Study: Helpsteer2	169
C.10.1	Analysis of Helpsteer2 Instances	169
C.10.2	Training the PPM on Helpsteer2	172
C.10.3	Routing instances in the Helpsteer2-Preferences dataset	172
C.11	Prompt Templates for Synthetic Preferences	173
C.11.1	Helpsteer2 prompt template	173
C.11.2	MULTIPREF prompt template	174
C.11.3	ChatArena and AlpacaFarm prompt template	174
D	Appendix: Post-training LMs with Open Data	187
D.1	Instruction Datasets Details	187

D.2	Model Training Details and Compute	189
D.3	Evaluation Setups	189
D.4	Overview of All Automatic Evaluation Results	191
D.5	Human Evaluation Details	196
D.5.1	Setup	196
D.5.2	Inter-Annotator Agreement	196
D.6	Further Investigation of Figure 5.2	198
D.7	Model Licenses	198

List of Figures

2.1	Example task from SUPERNI	28
2.2	A comparison of SUPERNI to other datasets in terms of task types	30
2.3	Performance on SUPERNI grouped by task types	37
2.4	Scaling trends of models performance on SUPERNI	38
2.5	Human evaluation results vs. ROUGE-L metrics.	38
3.1	A high-level overview of SELF-INSTRUCT	44
3.2	Selected tasks from the generated instruction data using vanilla GPT3.	45
3.3	Illustration of the most common patterns in generated instructions	50
3.4	Similarity between generated instructions and their most similar seed instructions	50
3.5	Length distribution of the generated instructions	50
3.6	Performance of GPT3 model and its instruction-tuned variants, evaluated by human experts on our 252 user-oriented instructions	55
3.7	Human evaluation of SELF-INSTRUCT with different sizes of instructions.	57
4.1	Overview of HYPER framework	63
4.2	Feature representation of candidate datasets and their actual reward modeling performance as the training data	67
4.3	Actual RewardBench performance of the best configuration found given n simulated candidates.	70
4.4	Comparison between HYPER and random selection given different annotation budgets on various preference datasets.	71
5.1	An example from ShareGPT data	83

5.2	Win-rate scores of 13B models (trained on different datasets) given by GPT-4 strongly correlate with the average numbers of unique tokens in the model responses	91
5.3	Human acceptance rates for four evaluated models.	92
5.4	Human preference rates for three comparison pairs of models.	92
A.1	The crowdsourcing template we use to receive feedback on our collected tasks.	122
A.2	Crowdsourcing interface used for human assessment of our baselines.	123
A.3	Encoding task instruction with input.	125
A.4	ROUGE-L v.s. Accuracy for task types that are usually regarded as classification tasks. The trends of these two metrics are highly correlated with each other even if not overlapped. . . .	126
A.5	Distribution of SUPERNI tasks in terms of their task types, languages, and domains.	126
A.6	Tk -INSTRUCT’s per-task performance improvement over the best of two heuristic baselines on the 119 evaluation tasks of the English track.	136
A.7	Tk -INSTRUCT’s per-task performance improvement over the best of two heuristic baselines on the 35 evaluation tasks of the cross-lingual track.	137
B.1	Distribution of the ROUGE-L scores between seed instructions and their most similar instructions in SUPERNI and the 252 user-oriented instructions	140
B.2	Human evaluation in done using a Google sheet, with predictions from different models present in random order and the model information being anonymized.	146
C.1	Construction of MULTIPREF involves three stages: data preparation, response generation, and human annotation.	152
C.2	Template for classifying the domain of a prompt.	153
C.3	Screenshot of our annotation platform	154
C.4	Gain distribution in MULTIPREF where gain is defined as the improvement in RM performance if a particular instance is routed to humans.	164
C.5	Proportion of prompts routed to humans or GPT-4 that belong to a specific level of expertise.	170
C.6	Top ten subjects of expertise needed to annotate instances for a subset routed to GPT-4 and subset routed to Humansin Helpsteer2.	170

C.7	Comparison between HYPER and random selection given fixed annotation budgets. We report the average of the RewardBench score across three runs.	171
C.8	Helpsteer2 prompts related to the <i>Computer sciences</i> subject but with different levels of expertise.	171
C.9	Comparison between HYPER and a random selection given different annotation budgets on the Helpsteer2-Preferences dataset	172
C.10	Disagreement between human and GPT-4 due to subjectivity in the user instruction.	173
C.11	Disagreement between human and GPT-4 due to choosing different responses even if both are correct.	174
C.12	Disagreement between human and GPT-4 due to the latter being wrong.	175
C.13	Disagreement between human and GPT-4 due to open-ended text generation request with high subjectivity.	176
C.14	Disagreement between human and GPT-4 due to the type of request.	177
C.15	Helpfulness prompt for Helpsteer2	178
C.16	Correctness prompt for Helpsteer2	179
C.17	Coherence prompt for Helpsteer2	180
C.18	Complexity prompt for Helpsteer2	181
C.19	Verbosity prompt for Helpsteer2	182
C.20	System prompt for MULTIPREF	183
C.21	Judgment prompt for MULTIPREF	184
C.22	System prompt for ChatArena and AlpacaFarm	185
C.23	Judge prompt for ChatArena and AlpacaFarm	186
D.1	The website interface for our human evaluation.	197
D.2	Win-rate scores of all models judged by the dummy evaluator against win-rate of all models using the GPT-4 evaluator.	198

List of Tables

2.1	A comparison of SUPERNI to other notable datasets	29
2.2	Statistics of SUPERNI.	33
2.3	Performance of different methods on SUPERNI	36
2.4	Performance of models trained and evaluated with various encodings.	39
3.1	Statistics of the generated data by applying SELF-INSTRUCT to GPT3.	51
3.2	Data quality review for the instruction, input, and output of the generated data.	52
3.3	Evaluation results on <i>unseen</i> tasks from Super-NI.	54
4.1	MULTIPREF dataset statistics.	69
4.2	Performance of the performance prediction model	70
4.3	Comparison of full direct human preferences and synthetic preferences and the best hybrid preference mix given unlimited budget on RewardBench.	72
4.4	Comparison of full direct human preferences and synthetic preferences on the best hybrid preference mix given unlimited budget on RewardBench and different base models of Helpsteer2	73
4.5	Comparison of full direct human preferences and synthetic preferences on the best hybrid preference mix given unlimited budget using Best-of-N evaluation.	74
4.6	Average gain in MULTIPREF’s performance when routing 100 random preference instances to a human annotator for each tag.	75
5.1	Instruction datasets investigated in the TULU work.	82
5.2	Base models that we finetuned in this work.	82
5.3	Comparison of different instruction tuning datasets	87

5.4	Performance of different base models after training on the Human+GPT data mixture.	87
5.5	Performance of TULU and other of our trained models to vanilla LLAMA models and the state-of-the-art proprietary models	89
5.6	Performance of models on ToxiGen and TruthfulQA	90
5.7	Win-rate of LLAMA models of varying sizes finetuned on the given dataset against InstructGPT ₀₀₃ using AlpacaEval	91
A.1	12 Evaluation categories, their evaluation metrics, and all the tasks in each category	128
A.2	An example task in the Textual Entailment category of our dataset, adopted from RTE	129
A.3	An example task in the Cause Effect Classification category of our dataset, adopted from COPA	129
A.4	An example task in the Cause Effect Classification category of our dataset, adopted from WinoGrande	130
A.5	An example task in the Dialogue Act Recognition category of our dataset, adopted from MetaLWOz	130
A.6	An example task in the Answerability Classification category of our dataset, adopted from AdversarialQA	131
A.7	An example task in the Word Analogy category of our dataset, adopted from BARD	131
A.8	An example task in the Overlap Extraction category of our dataset, adopted from PointsOfCorrespondence	132
A.9	An example task in the Keyword Tagging category of our dataset, adopted from OHSUMED	133
A.10	An example task in the Question Rewriting category of our dataset, adopted from Disfl-QA .	133
A.11	An example task in the Title Generation category of our dataset, adopted from PerSenT . . .	134
A.12	An example task in the Data to Text category of our dataset, adopted from E2E	135
A.13	An example task in the Grammar Error Correction category of our dataset, adopted from JFLEG	135
B.1	Hyper-parameters for querying OpenAI API in different experiments.	140
B.2	Prompt used for generating new instructions.	141
B.3	Prompt used for classifying whether a task instruction is a classification task or not.	142
B.4	Prompt used for the input-first approach of instance generation.	143

B.5	Prompt used for the output-first approach of instance generation	144
B.6	Examples in the user-oriented instructions dataset and predictions from GPT3 _{SELF-INST.}	147
B.7	Representative valid tasks generated by GPT3.	149
B.8	Representative invalid tasks generated by GPT3.	150
C.1	Number of prompts in MULTIPREF taken from each source.	153
C.2	Qualification results for normal and expert crowdworkers, and the number of prompts per domain.	154
C.3	Qualification test for recruiting potential annotators.	161
C.4	RewardBench scores of reward models using different inference-time sampling strategies based on a linear model: top- k and simulated.	162
C.5	RewardBench scores of reward models using different inference-time sampling strategies based on a quadratic model: top- k and simulated.	162
C.6	Lexical and descriptive tags obtained from the prompt-response triples $\langle x, y_1, y_2 \rangle$ in order to find a subset $S \subset D$ to route to human annotators.	163
C.7	Average gain in MULTIPREF’s performance (as predicted by the quadratic regressor) when routing random 100 units to human annotators.	165
C.8	Finegrained RewardBench results on the Chat category.	167
C.9	Finegrained RewardBench results on the Chat-Hard category	167
C.10	Finegrained RewardBench results on the Safety category	167
C.11	Finegrained RewardBench results on the Reasoning category	167
C.12	Comparison of DPO-trained models using different human-LLM preference mixes.	168
C.13	Reward Model Training Hyperparameters	169
C.14	Performance comparison between Helpsteer2 and Helpsteer2-Preferences on RewardBench for different annotation budgets.	175
D.1	An overview of the performance of all models finetuned for this work, along with proprietary models, on selected benchmarks.	192
D.2	ToxiGen results across models.	194

D.3 TruthfulQA results across models. 195

Chapter 1

Introduction

1.1 Brief Overview

The rapid advancement of artificial intelligence (AI) has been significantly propelled by the development of large language models (LLMs, [Devlin et al., 2019](#); [Radford et al.](#); [Raffel et al., 2020](#); [Brown et al., 2020a](#)). These models, pretrained on vast quantities of text data, exhibit remarkable general-purpose capabilities, having encoded extensive knowledge about the world. However, harnessing this latent potential and effectively directing these models to serve a multitude of downstream applications—ranging from following complex instructions and engaging in nuanced conversations to utilizing external tools and performing sophisticated reasoning—remains a critical challenge. This dissertation addresses this challenge by proposing and investigating scalable data paradigms as a primary mechanism for steering general-purpose language models. The core thesis is that by systematically rethinking how we structure, create, share, and utilize data in the post-training phase (as a counterpart to pretraining), we can more effectively and efficiently align LLMs to diverse human intents and complex tasks.

1.2 Background

General-purpose language models, often built upon transformer architectures and pretrained on internet-scale unlabeled text, have become foundational in modern AI [[Bommasani et al., 2021](#)]. This pretraining phase endows them with a broad understanding of language, factual knowledge, and rudimentary reasoning skills.

However, their raw pretrained state is often insufficient for specific, practical applications. To bridge this gap, a crucial phase known as “post-training” (or “alignment” or “instruction tuning”, [Ouyang et al., 2022](#); [Bai et al., 2022a](#); [Touvron et al., 2023a](#); [OpenAI, 2023](#)) is employed. This phase aims to steer the model’s behavior to better align with human expectations and desired functionalities. This dissertation focuses on data-centric approaches within this post-training landscape, exploring how innovative data paradigms can unlock and refine the capabilities of these powerful general-purpose systems.

1.3 Challenges

Steering general-purpose LMs effectively presents several key challenges. Firstly, the diversity of tasks and interaction styles humans expect LMs to handle is immense. Crafting datasets that comprehensively cover this spectrum is a monumental undertaking. Secondly, traditional methods of data collection and annotation can be expensive, time-consuming, and may not scale to meet the ever-growing demands for more capable and nuanced models. Thirdly, ensuring that models generalize well from the data they are post-trained on, without simply memorizing specific task formats, is crucial for robust real-world performance. Furthermore, as models become more integrated into data production and annotation processes, new questions arise about data quality, diversity, and the potential for reinforcing biases. The diminishing returns observed from simply scaling pretraining on raw internet data also highlight the urgent need for more sophisticated and sustainable data strategies for the next generation of AI systems.

1.4 Approach: Scalable Data Paradigms

This dissertation posits that addressing the aforementioned challenges requires a fundamental shift towards **scalable data paradigms**. Instead of viewing data as a static resource, we explore dynamic and systematic approaches to data creation, curation, and utilization specifically tailored for steering LLMs. My research investigates four such paradigms:

- **Unified Data Representation:** This paradigm focuses on unifying the way different NLP tasks are presented to models. By leveraging natural language instructions as a common interface, we can enable models to generalize across a broader range of tasks.

- **Synthetic Data Generation:** This paradigm explores the use of LMs themselves in the data production process. By developing frameworks where models can generate new training instances and even novel task types, we can augment and diversify training data at scale, potentially leading to model self-improvement.
- **Human-AI Collaborative Data Annotation:** Recognizing the strengths and weaknesses of both human and machine annotation, this hybrid paradigm seeks to optimize the data annotation process. By intelligently routing annotation tasks between humans and AI systems, we aim to achieve high-quality data more efficiently and cost-effectively.
- **Open Data and Models for Post-Training:** This paradigm emphasizes the importance of open resources, including datasets and models, in advancing research and democratizing the development of capable LLMs. It involves systematically studying how diverse open datasets contribute to model capabilities during post-training and fostering collaborative efforts.

Together, these data-centric strategies aim to create a more generalizable, efficient, and collaborative foundation for developing and steering the next generation of general-purpose language models, moving beyond simple pretraining to more nuanced and effective post-training.

1.5 Dissertation Outline

The remainder of this dissertation is structured as follows:

Chapter 2: Unifying NLP Tasks by Adding Instructions delves into the unified data paradigm. It introduces the SUPER-NATURALINSTRUCTIONS benchmark [Wang et al., 2022b], a large-scale meta-dataset of NLP tasks with natural language instructions. This chapter details the construction of this dataset and presents Tk -INSTRUCT, a model trained on it that demonstrates strong generalization to unseen tasks by following instructions, outperforming larger models in some cases. This work underscores the power of redefining task data structure as an instruction tuning format and enhancing model generalization.

Chapter 3: Synthesizing Training Data with Models explores the synthetic data paradigm. It introduces SELF-INSTRUCT, a framework that enables language models to generate their own instruction-following data with minimal human seed data [Wang et al., 2022a]. This chapter details the iterative bootstrapping process

of SELF-INSTRUCT, the characteristics of the generated data (52k instructions), and demonstrates how fine-tuning a model on this self-generated data can significantly improve its instruction-following capabilities and even enable model self-improvement.

Chapter 4: Routing Annotations between Human and AI focuses on the *human-AI collaboration paradigm* for data annotation, specifically for preference data used in reinforcement learning from human feedback (RLHF). It introduces HYPER(Hybrid Preference Router), a method that intelligently allocates preference annotation tasks to either human annotators or LMs to optimize the cost-efficiency and quality of the resulting dataset [Miranda et al., 2024]. The chapter details the MULTIPREF dataset created for this research and shows how HYPER can lead to better reward models compared to using purely human or purely synthetic preferences.

Chapter 5: Post-training LMs with Open Data investigates the open data paradigm. This chapter details a systematic study of post-training LLMs using a wide array of publicly available instruction-tuning datasets. It analyzes how different datasets contribute to various model capabilities and the importance of base model quality. The work highlights the development of TULU 1 [Wang et al., 2023b], an open model that demonstrates strong performance by leveraging a curated mixture of open data resources, emphasizing the role of data curation and open collaboration in advancing AI. The TULU series of work has been continuously developed [Iverson et al., 2023, 2024; Lambert et al., 2024a], as part of larger team work.

Finally, Chapter 6 will summarize the key contributions of this dissertation and discuss promising directions for future research in steering general-purpose language models through scalable data paradigms.

Chapter 2

Unifying NLP Tasks by Adding Instructions

2.1 Overview

The NLP community has witnessed great initial progress in building models for generalization to *unseen* tasks via in-context instructions [Mishra et al., 2022b; Sanh et al., 2022; Wei et al., 2022a] using large pretrained language models [Raffel et al., 2020; Brown et al., 2020b]. As remarkable as models like InstructGPT [Ouyang et al., 2022] are, the contribution of various design choices to their success is opaque. In particular, the role of supervised data has remained understudied due to limited data released by the corporate entities behind major models. In addition, it is nearly impossible for the research community to extend and re-train these gigantic models. Addressing these two challenges necessitates the availability of large-scale public benchmarks of a broad range of NLP tasks and their instructions to facilitate developing and evaluating models that can generalize to unseen tasks.

In this chapter, we describe our efforts on constructing a meta-dataset (i.e., dataset of datasets; Triantafillou et al., 2019) that consists of a wide variety of NLP tasks with their instructions, and train a model that can perform a new task given the instruction, outperforming InstructGPT (which uses $16\times$ more parameters).

Our dataset, SUPER-NATURALINSTRUCTIONS (SUPERNI for short), is a large benchmark of 1,616 NLP tasks and their natural language instructions. It brings in a diverse variety of tasks—76 broad task types spanning 55 different languages. Each task is paired up with an instruction that consists of the task definition for mapping an input text to a task output and several examples for demonstrating the desired

or undesired output (see Fig.2.1 as an example task). These tasks and their instructions are contributed by 88 NLP practitioners, in response to our public call. These contributions are consolidated after several rounds of peer-review and crowdsourced feedback to ensure quality. Having this diverse and large-scale data enables us to carefully split the tasks into training and test sets and systematically study how state-of-the-art methods perform on them. Table 2.1 highlights properties of SUPERNI compared to relevant benchmarks, emphasizing the diversity of tasks and instruction types in our benchmark.

Our model, Tk -INSTRUCT, is a generative model for transforming task inputs given declarative in-context instructions (task definition or k -shot examples). It is built by multi-task training of the T5 model [Raffel et al., 2020] over all the task instructions in our training set, and is evaluated on unseen tasks in the test set. Interestingly, an 11B-parameter Tk -INSTRUCT can outperform the 175B-parameter InstructGPT model by 9.9 ROUGE-L points when evaluated on 119 unseen English tasks, and the multilingual variant mTk -INSTRUCT outperforms InstructGPT by 13.3 points on 35 non-English tasks (§2.5).

The compelling empirical performance of Tk -INSTRUCT confirms the importance of super-sized meta datasets such as our SUPERNI to facilitate research towards generalizable NLP models. We conduct extensive analysis to understand the important factors for this generalization (§2.6). Our analysis shows that scaling up the diversity of training tasks and the model size are both important for strong generalization to unseen tasks. Finally, we estimate performance upper bounds, suggesting further room for improvement.

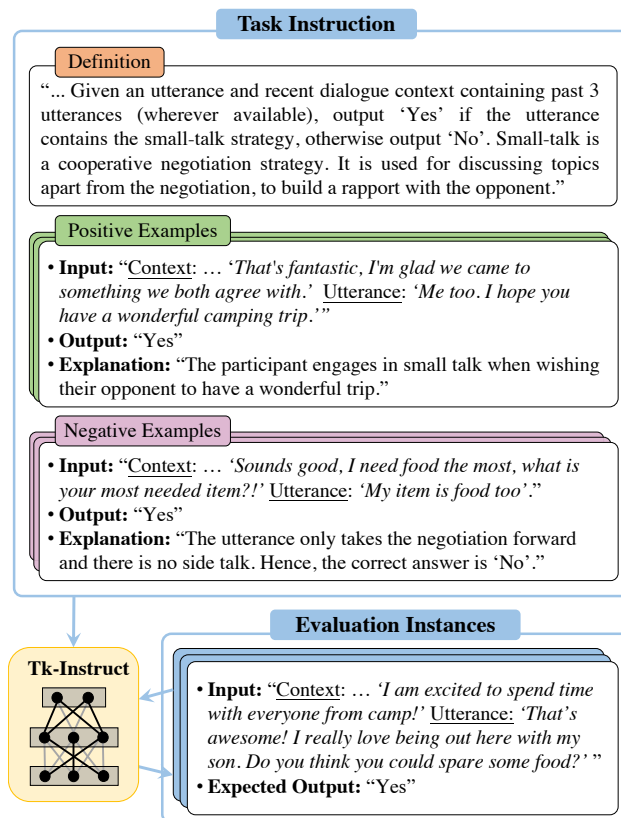


Figure 2.1: An example task from SUPERNI adopted from Chawla et al. [2021]. A successful model is expected to use the provided instructions (including task definition and demonstration examples) to output responses to a pool of evaluation instances.

Resource →	SUPERNI (this work)	NATINST Mishra et al. [2022b]	CROSSFIT Ye et al. [2021a]	PROMPTSOURCE Bach et al. [2022]	FLAN Wei et al. [2022a]	GPT3 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	–

Table 2.1: A comparison of SUPERNI to a few notable datasets in the field. We obtain the number of tasks, instructions, and task types of other datasets from their original paper. “–” indicates the fields are not applicable or unknown.

2.2 Background

Language instructions are a versatile way of defining goals, which is why they have been studied in the context of a variety of applications. Prior work has studied “instructions” in various niches, such as instructions in grounded environments [Shridhar et al., 2020; Stepputtis et al., 2020; Min et al., 2022b; Weir et al., 2022] and database commands [Kim et al., 2020]. Here, we focus on applications of instructions for general NLP tasks.

Recent literature has been motivated by building models that are generalizable across a variety of NLP tasks, when prompted with either a few examples [Ye and Ren, 2021; Bragg et al., 2021] or language definitions [Efrat and Levy, 2020; Weller et al., 2020; Zhong et al., 2021; Mishra et al., 2022b,a]. Our work is related to the existing benchmarks in this line of work, as delineated in Table 2.1 along various dimensions. Our benchmark extends NATINST [Mishra et al., 2022b] with $26\times$ more tasks and more variety of task types (Fig. 2.2). ZEST [Weller et al., 2020] focuses on tasks with question-like task descriptions that are generally shorter. Compared to this work, our instructions are longer, more complex and natural since they were used to collect datasets through crowdsourcing. While CROSSFIT [Ye et al., 2021a] focuses on benchmarking with a few in-context examples, our benchmark also offers task instructions.

Concurrent to our work, PROMPTSOURCE [Bach et al., 2022] is another benchmark of tasks and their language instructions (prompts). An important distinction between this benchmark and ours is the phrasing of the task definitions: while PROMPTSOURCE task definitions are relatively concise, our task definitions are collected with the intention of providing a complete definition of each task and therefore are longer (24 tokens vs. 56 tokens on average; Table 2.1). More recently, BigBench [Srivastava et al., 2022] introduces a collection of 204 tasks that are challenging for current LMs, though not focusing on collecting their instructions.

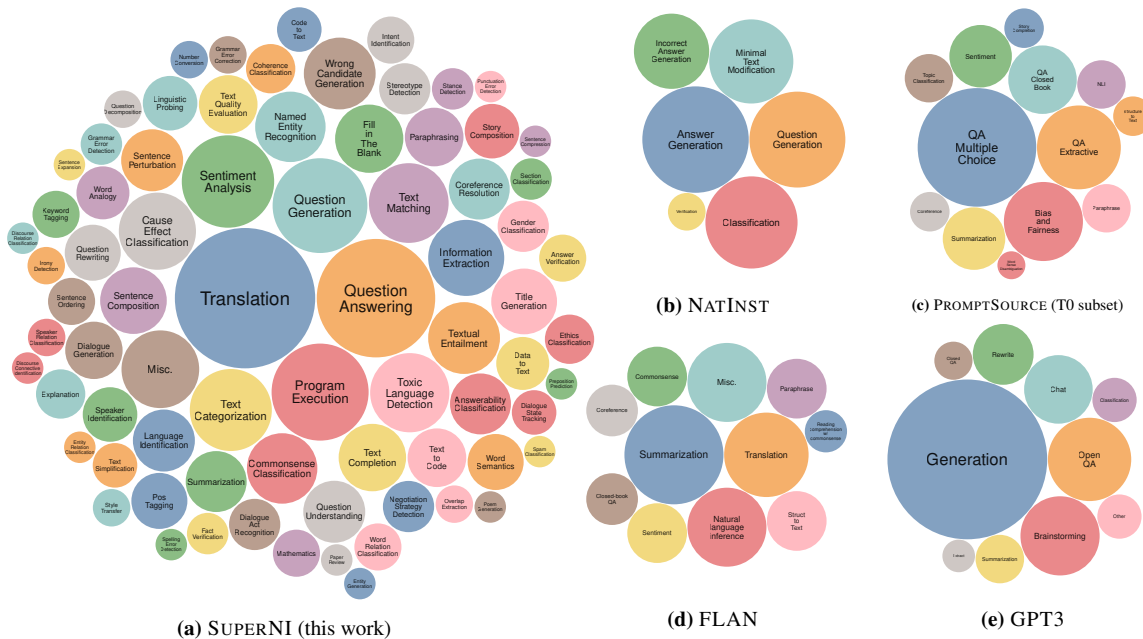


Figure 2.2: Compared to other datasets, SUPERNI covers a more diverse range of task types. InstructGPT reports a very coarse categorization of their task types. Bubble size represents the number of tasks of each type in log scale.

Nevertheless, we believe that all these efforts in collecting different tasks as well as the task instructions are complementary, and the community will benefit from considering different benchmarks. Finally, the well-adopted InstructGPT model [Ouyang et al., 2022] is partially enabled by a large dataset of prompts that are collected via various synthetic data augmentation which, unfortunately, is not publicly available.

Beyond cross-task generalization, our benchmark can also be used to study multi-task learning more broadly, which is a longstanding goal for AI [Caruana, 1997]. Traditionally, this literature focuses on setups that involve evaluation on tasks that are observed during training [Collobert and Weston, 2008; Hashimoto et al., 2017]. More recent studies show promise that large-scale multi-task learning can enable strong generalization to similar tasks via unified encoding [Khashabi et al., 2020; Xie et al., 2022] or better finetuning results on downstream tasks [McCann et al., 2018; Aribandi et al., 2022]. Our proposed benchmark provides diverse tasks for studying multi-tasking at a massive scale.

2.3 SUPER-NATURALINSTRUCTIONS dataset

SUPER-NATURALINSTRUCTIONS (SUPERNI for short) is a meta-dataset [Triantafillou et al., 2019] consisting of a variety of NLP tasks (see Fig. 2.2a) and instructions that describe them in plain language.

2.3.1 Instruction schema

All task instructions follow the same uniform schema (see Fig. 2.1) which is composed of the following parts:

- DEFINITION defines a given task in natural language. This involves a complete definition of how an input text (e.g., a sentence or a document) is expected to be mapped to an output text.
- POSITIVE EXAMPLES are samples of inputs and their *correct* outputs, along with a short explanation for each.
- NEGATIVE EXAMPLES are samples of inputs and their *incorrect/invalid* outputs, along with a short explanation for each.

The above schema is based on that of Mishra et al. [2022b], though it is simplified. See Appendix A.3 for the comparison.

Task instances Given the instructions for each task, a model is expected to solve instances of that task. We use a unified format to organize the instances of all our tasks. More precisely, each instance consists of a textual input and a list of acceptable textual outputs. These instances are either drawn from existing NLP datasets or available sources of human annotations. We limit the number of instances in each task to 6.5K to avoid an imbalance of instances between tasks.

2.3.2 Collection process

SUPERNI was collected through a large community effort on GitHub¹. Tasks were collected and contributed by NLP practitioners who were either responding to our public invitation² or students who were encouraged to contribute as part of their class project³. Contributors were encouraged to be creative and source the

¹<https://github.com/allenai/natural-instructions>

²<https://blog.allenai.org/9d3f24d5a9db>

³CSE 576 “Topics in NLP” course, Arizona State Univ.

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, or finding the longest palindrome substring). When using existing datasets or crowdsourcing annotations, contributors were encouraged to adopt the instructions used to create this dataset, whenever available. This was done to ensure that the instructions were sufficient to define the tasks to average human readers. Tasks along with instructions and other meta information were contributed as JSON files via GitHub pull requests, which were reviewed by automated checks and peers. 88 contributors from diverse locations and backgrounds contributed to our repository.

Controlling the quality of this community-contributed data was done in several phases: (1) Upon creating a GitHub pull request of the proposed task, it immediately went through an automatic test. This process verified that the introduced file contained the expected fields and adhered to our desired properties (e.g., no duplicate instances, the output labels are not heavily imbalanced, etc.). (2) The proposed task was then peer-reviewed by 1~2 other expert contributors to ensure the clarity and sufficiency of instruction content. The review process was done iteratively until the reviewers were content with the quality of the proposed instruction. Specifically, reviewers were asked to verify if the instruction is clear and sufficient for an average language speaker to solve the underlying task (evaluation instances) while being grammatical, fluent, and concise. On average, the review of each GitHub pull request took about 4-6 iterations over the span of multiple days before merging them. (3) Lastly, the added tasks were presented to crowdworkers in order to collect feedback on the quality of the provided instructions, such as typos, clarity, or other issues. (details in §A.1). Subsequently, one of the authors use this feedback to improve the task definitions of their instances. This feedback was done only for English tasks, as finding high-quality crowdworkers in other languages is nontrivial [Pavlick et al., 2014].

2.3.3 Diversity of tasks

Collecting tasks for SUPERNI was supervised carefully to cover a variety of natural language understanding tasks, domains, and languages. To better understand this diversity, we comprehensively categorize tasks along three different dimensions:

- TASK TYPE defines the nature of the mapping from instance inputs to outputs (e.g., question answering, classification, etc.).
- LANGUAGE indicates the language(s) of the instances.
- DOMAIN indicates the domain(s) to which the language of the tasks belong to (e.g., politics, medicine, dialogue, etc.).

These different measures of categorization can be used to study different senses of generalization. In our empirical studies (§2.5), we study generalization along the axis of task types. We refer the reader to Fig. A.5 in Appendix for the distribution of tasks among different task types, languages and domains.

Table 2.2 shows various statistics for the benchmark. In total, the dataset includes 1616 tasks and 5M instances. On average, each instruction is paired with 2.8 positive and 2.4 negative examples. The average definition length is 56.6 in words.

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 2.2: Statistics of SUPERNI.

2.4 Teaching Models to Follow Instructions

2.4.1 Formulation

Each task t is defined via its natural language instruction I_t , and each task has a set of input/output instances (X_t, Y_t) . A model M is expected to produce the output y , given the input x and the task instruction I_t : $M(I_t, x) = y$, for $(x, y) \in (X_t, Y_t)$. In particular, we would like to evaluate model M on tasks that are not

observed (i.e., their instances were not used for training M). The only source of signal for learning the task at inference time is in-context instructions I_t that contains a definition and demonstration examples of the task.

2.4.2 T_k -INSTRUCT

We introduce T_k -INSTRUCT that is meta-trained on SUPERNI for solving tasks given their in-context instructions. Previous work has shown the effectiveness of such kind of meta-training in improving model’s ability to do in-context learning with either prompts [Zhong et al., 2021; Sanh et al., 2022] or demonstration examples [Min et al., 2022a]. Because of the large variety of tasks in SUPERNI, we are able to do this multi-task meta-training at an ever-large scale. We conduct our experiments and analysis based on the T5 model [Raffel et al., 2020]. Since each instruction I_t consists of multiple elements as described in our instruction schema (§2.3.1), we map these elements to textual format and append them before the input instance. Fig. A.3 in Appendix shows how we encode the full instruction. We study different combinations of these instruction elements in §2.6.2. By default, we will use our most effective instruction elements (i.e., task definition and two positive examples) unless otherwise specified. In the same manner, we train the multilingual variant mT_k -INSTRUCT based on the mT5 model [Xue et al., 2021].

2.5 Results

2.5.1 Benchmarking Setup

We provide a recommended recipe for benchmarking generalization via SUPERNI.

An Evaluation Split of Unseen Tasks. We split the large collection of tasks in SUPERNI into two subsets: one for evaluation and the other for supervision. For evaluation tasks, we fix a manually-selected collection of 12 categories that represent 154 tasks. The large variety of tasks in SUPERNI enables us to choose a diverse set of tasks for evaluation – such as those at word, sentence, and document levels, covering both classification and generation formats. Appendix A.7 lists our evaluation tasks with examples for representative tasks. For an efficient evaluation, we sample a maximum of 100 instances for each task, which results in 15,310 testing

instances in total. The remaining tasks are used for training models.⁴.

Divided Tracks for English and X-lingual Tasks SUPERNI consists of tasks across multiple languages, which enables evaluating the model’s generalization to unseen tasks not only in English but also in other languages. Therefore, we divide our evaluation tasks into two tracks: one for **English cross-task generalization** (119 tasks), and the other for **cross-lingual cross-task generalization** (35 tasks). To the best of our knowledge, this is the first study in cross-lingual cross-task generalization (i.e., generalization to unseen tasks in different languages). Fig. A.6 and Fig. A.7 in the appendix contains the evaluation tasks for each track.

Evaluation Metrics Due to the diversity of our tasks and the open-ended generation nature of our formulation⁵, we adopt *ROUGE-L* [Lin, 2004] for reporting aggregated performance results. This is a soft string overlap metric that can be applied to a wide range of text generation tasks. We show that the ranking from this metric correlates well with accuracy for classification tasks in Appendix A.5. We also conduct human evaluation in §2.5.4.

2.5.2 Baselines and Existing Models

Here we discuss a variety of baselines and competitive models for our target application. See Appendix A.4 for implementation details.

Heuristic baselines. We first evaluate the following heuristics to evaluate the possible shortcuts in the data. **Copying Demo Output** copies the output of a random demonstration example. Since we balance the labels for our test tasks, the performance of this baseline will roughly equal a random guess or a majority baseline for classification tasks. **Copying Instance Input** copies the given instance input. This strategy performs well on tasks where the target output largely overlaps with the input (e.g., question rewriting, grammar error correction).

⁴To avoid data leakage, we exclude tasks from the training set if they are sourced from the same dataset as any test task. This results in 757 training tasks for the English track and 1271 training tasks for the X-lingual Track

⁵Unlike Sanh et al. [2022]; Wei et al. [2022a] who evaluate their models on classification tasks via option ranking (i.e., scoring the correct answer(s) higher than other candidate answers), we evaluate our models in an open-ended generation setting with no task-specific assumptions. We believe this is a more realistic measure of generalization to unseen tasks.

	Methods ↓ / Evaluation →	English	X-lingual
Heuristic Baselines	Copying Instance Input	14.2	5.4
	Copying Demo Output	28.5	50.3
Pretrained LMs	T5-LM (11B)	30.2	–
	GPT3 (175B)	45.0	51.3
Instruction-tuned Models	T0 (11B)	32.3	–
	InstructGPT (175B)	52.1	52.8
	<i>Tk</i> -INSTRUCT (ours, 11B)	62.0	–
	<i>mTk</i> -INSTRUCT (ours, 13B)	57.1	66.1
Upper-bound (est.)	Supervised Training	74.3	94.0

Table 2.3: The overall performance of different methods on unseen tasks in the test set of SUPERNI (§2.5.3). We report ROUGE-L here as our aggregated metric.

Off-the-shelf pretrained language models. We evaluate existing LMs that are not fine-tuned with instruction-specific data. Specifically, we evaluate T5-11B [Raffel et al., 2020], the “LM-adapted” version that is further trained with language model objective [Lester et al., 2021]. Additionally, we evaluate GPT3 [Brown et al., 2020b], a 175B-parameter autoregressive LM that has shown remarkable ability in following demonstrations provided in its prompt.

Instruction-tuned models. In addition to our *Tk*-INSTRUCT (§2.4), we evaluate existing models that are fine-tuned to follow language instructions. In particular, we evaluate InstructGPT [Ouyang et al., 2022] which uses reinforcement learning to incorporate human preferences into a GPT3 pretrained model, and T0 [Sanh et al., 2022] which finetunes T5 on a collection of task prompts in PROMPTSOURCE [Bach et al., 2022].

Upper bound estimates. We estimate an upper bound on models’ generalization to unseen tasks by fine-tuning an **oracle** model on the tasks’ labeled instances. Since this model observes the *hidden* instances of the evaluation tasks, it is, by definition, an estimated upper bound to our generalization-based models. Specifically, we fine-tune a T5-11B model on the 119 English evaluation tasks, and a mT5-13B model on the 35 non-English tasks, with 1K training instances per task. Since this model observes the evaluation tasks, it is, by definition, an upper bound to our generalization-based models.

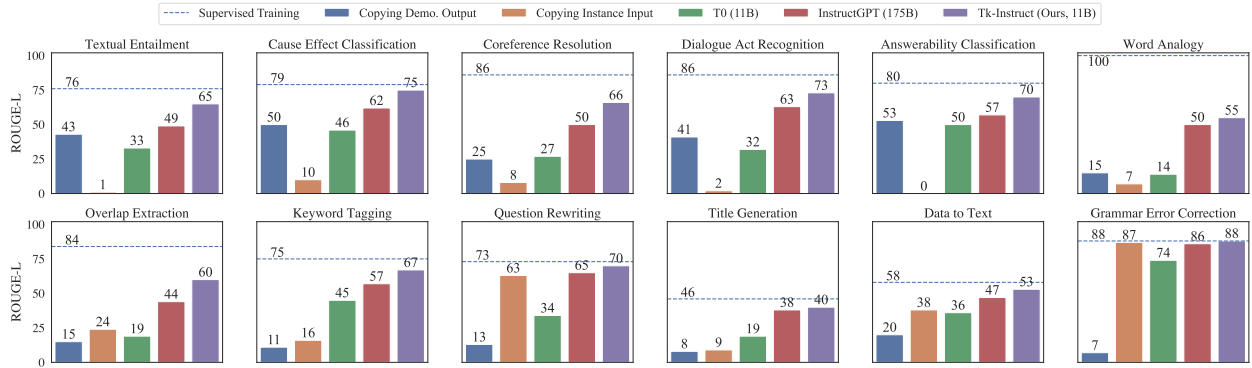


Figure 2.3: Performance per evaluation task type. *Tk*-INSTRUCT consistently performs better than other generalization-based methods on all task types, while there is still a sizable gap compared to supervised training.

2.5.3 Main Results

Table 2.3 summarizes our overall benchmarking results. We use the same input encoding that contains the most effective instructional elements (task definition and two positive examples) for all the methods. To better understand models’ generalization to different tasks, we also break down the performance according to the task categories in Fig. 2.3. We refer the reader to Appendix A.8 for more detailed analysis on each individual task.

Instruction-tuning enables stronger generalization to unseen tasks. Generally instruction-tuned models perform remarkably better compared to their untuned LM counterparts (*Tk*-INSTRUCT v.s. T5-LM, InstructGPT v.s. GPT3) and heuristic baselines. This indicates models do learn to follow instructions by finetuning on instruction data, and this can generalize to new instructions for unseen tasks. T0 is an exception, which is only slightly better than T5-LM. We suspect this is because the style of prompting in T0’s training data is very different from our style of instructions.

Our *Tk*-INSTRUCT outperforms InstructGPT. Our *Tk*-INSTRUCT and *mTk*-INSTRUCT models, which are trained with a variety of tasks, generalize best to unseen tasks for both English and non-English tasks in all evaluation task categories. InstructGPT also shows a great extent of generalization to our evaluation tasks. However, we want to note it is not clear if InstructGPT’s training data overlaps with our evaluation tasks since their data is unavailable.

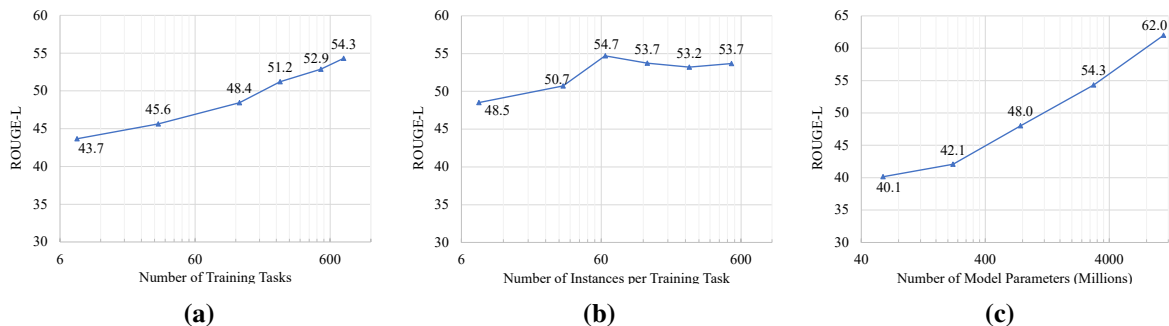


Figure 2.4: Scaling trends of models performance (§2.6.1) as a function of (a) the number of training tasks; (b) the number of instances per training task; (c) model sizes. x -axes are in log scale. The **linear growth of model performance with exponential increase in observed tasks and model size** is a promising trend. Evidently, the performance gain from more instances is limited.

There is a sizable gap for improvement. Despite the impressive performance of current models, there is a sizable gap between the generalization of instruction-based models and the supervised training approach, leaving more room for improvement.

2.5.4 Human Evaluation

We additionally perform human evaluation which is generally known to provide more accurate evaluation, especially for open-ended generation tasks. Specifically, we ask crowdworkers to indicate if they prefer the predicted answer by the model or the ground truth outputs for each instance (see Appendix A.2 for details). The resulting human evaluation metric indicates how often were model predictions rated as at least as good as our ground truth

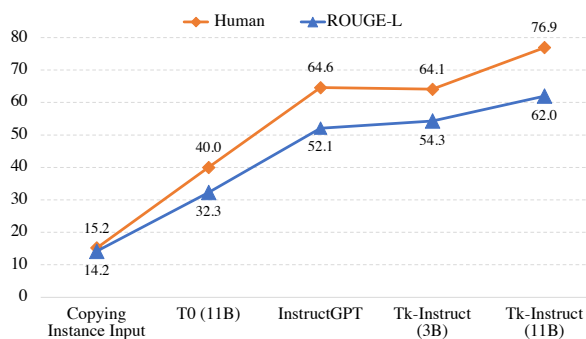


Figure 2.5: Human evaluation vs. ROUGE-L (§2.5.4).

labels. The theoretical upperbound of this metric is 100% when the model is rated at least as good as the ground truth for all the instances. The results of human evaluation (shown in Fig. 2.5) align quite well with our automatic metrics and confirm the human-perceived quality of our models.

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	<u>21.2</u>	33.3	16.8	30.9	23.0	33.7	33.9	31.6	26.0	36.4	33.9
Def	17.3	<u>45.0</u>	31.1	43.8	36.4	46.4	44.2	44.3	38.0	46.0	39.9
Pos (1)	10.9	22.1	<u>43.9</u>	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	<u>52.4</u>	47.4	53.3	53.1	51.8	47.8	53.7	44.5
Pos (2)	12.7	22.4	47.1	50.2	<u>49.3</u>	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	<u>54.3</u>	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	<u>54.3</u>	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	<u>52.6</u>	48.6	52.2	44.3
Pos (4)	11.0	23.9	45.6	49.8	49.0	51.7	49.5	47.5	<u>49.8</u>	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	<u>53.5</u>	46.0

Table 2.4: Performance (ROUGE-L) of models trained and evaluated with various encodings. Diagonal numbers (underlined) represent performances of models trained and evaluated with the same instruction encoding. Each encoding is a combination of the elements in the instructions (Fig. 2.1). **Task ID** is a short string composed of dataset name and task category; **Def** represents the task definition; **Pos (k)** represents k positive examples; **Neg (k)** represents k negative examples; **Expl** represents explanation. These results (a) show the gains from various instructional elements, and (b) indicate surprising reliability of the models to various input encoding. Specifically, a model trained with definition and positive examples (e.g., the last row) remains robust for different encodings.

2.6 Further Analysis

We conduct further analysis to understand the important factors for models to generalize across tasks. Due to the computational cost, this analysis is done on the English track and using the T5-3B checkpoint, except for the experiments on model sizes.

2.6.1 Scaling Trends of Generalization

We study Tk -INSTRUCT’s generalization performance with respect to three scaling factors: the number of training tasks, the number of instances per task, and the model sizes. Fig. 2.4 presents the performance change by scaling each of them.

More observed tasks improve the generalization. We fine-tune Tk -INSTRUCT with different numbers of tasks that are randomly sampled from the whole training set (Fig. 2.4a). The model generalization performance grows log-linearly⁶ as we increase the set of tasks used for training. Previous work [Mishra et al., 2022b; Sanh et al., 2022; Wei et al., 2022a] has made similar observations on a much smaller scale, while we show that this trend holds even with 757 diverse training tasks.

⁶A linear function of an exponential increase of parameters, i.e., growth at a constant multiplicative rate.

A large number of training instances do not help generalization. We then vary the number of instances per task that are used for finetuning (Fig. 2.4b). While the conventional wisdom in supervised learning is that more training instances usually helps [Banko and Brill, 2001; Sun et al., 2017; Hestness et al., 2017], in our setup, the model’s performance saturates when only 64 instances per task are used for training. A large number of training instances would instead lead to longer training time and risk overfitting to the training tasks.

Tuning larger models with instructions consistently lead to gains. We study the effect of model scaling by initializing Tk -INSTRUCT from different sizes of pretrained T5 checkpoints, including the small, base, large, xl and xxl sizes (Fig. 2.4c). We found that increasing the model sizes consistently bring significant improvement (log-linearly with parameter size). This finding contradicts the claim in Xu et al. [2022] that “model size has little impact on performance with an extremely large number of tasks.” Combining Fig. 2.4 (a) and Fig. 2.4 (c), one can create a correspondence between model size and task size. For example, a T5-large model trained with 757 tasks can achieve comparable performance (48.0 ROUGE-L) to the T5-3B model trained with 128 tasks (48.4 ROUGE-L), indicating that increasing the diversity of training tasks is an alternative to scaling models sizes.

2.6.2 Instructing with Different Elements

We evaluate the performance of Tk -INSTRUCT under different instructional elements.

Benefit of different instructional elements. As shown in Fig. 2.1, SUPERNI provides multiple elements for instructing a task. We train multiple models with different combinations of these elements. The *diagonal* cells of Table 2.4 show the performance of our models when trained and evaluated on a particular instruction encoding. Based on the diagonal numbers, including the task definition is necessary for the model to successfully generalize. Moreover, combining the task definition with examples yields significant improvement. However, adding more demonstration examples is negligible. Negative examples help a little bit; explanations decrease performance, which is consistent with the observations of Mishra et al. [2022b] and Lampinen et al. [2022] when the model is not large enough. Future work can explore whether more powerful models can benefit from these elements.

Generalization to different input encodings. We further investigate whether a model trained on a particular encoding can generalize to other encodings. This can be read from the *non-diagonal* cells of Table 2.4. The negative result here is that definition-only models cannot generalize to example-only test encodings; and similarly, example-only models cannot generalize to definition-only test encodings either. However, models trained on encodings that contain both definition and examples are surprisingly robust across different encoding variations.

2.7 Discussion & Future Work

While the presented data offers a notable variety (e.g., diverse task types), its underlying distributions suffer from skews, which should be addressed in future work (see Appendix A.6). On language diversity, the proposed benchmark is biased toward English. On output diversity, the collected tasks are generally still skewed to short responses, reflecting which might reflect the distribution of the available tasks in the field. This under-presentation of the long-tail of tasks poses a challenge for building general-purpose models in the future. We hope future work addresses such distributional imbalances. Moreover, we see natural extensions of the instruction-following setup here in the context of other modalities such as vision or speech.

Automatic evaluation of models' performance is another challenge, considering the diverse set of tasks in our benchmark, and many of them being open-ended generation tasks. We use ROUGE-L as an aggregated metric in this paper and find it as a good proxy for the *overall* performance of the models, aligning well with human evaluation. However, there do exist specific tasks for which ROUGE-L might not serve as an effective proxy of quality (such as rewriting tasks or error correction tasks where copying the input can result in a high ROUGE-L score). We hope these issues will be addressed with the development of more powerful evaluation metrics for text generation.

In terms of computing power, we have tried to experiment with models that were accessible to us and we will make the resulting models publicly available so that the future readers do not need to spend energy repeating our work. We also acknowledge that there are larger models that we were not able to train due to the limitations on our computational budget.

2.8 Summary

We construct a large-scale benchmark consisting of a diverse set of NLP tasks and their instructions. This benchmark can serve as a rich playground for training or evaluation of models that can generalize to unseen tasks by following instructions. Furthermore, we train *Tk*-INSTRUCT using this data, and demonstrate its capability to perform unseen tasks to a surprising extent. We provide extensive analysis to understand the important factors for such generalization.

Chapter 3

Synthesizing Training Data with Models

3.1 Overview

As is introduced in the last chapter, instruction tuning is an effective approach for steering language models to generalize across many tasks. The success of instruction tuning is powered by two key components: large pretrained language models (LM) and human-written instruction data [Mishra et al., 2022b; Wei et al., 2022a; Sanh et al., 2022; Wang et al., 2022b; Ouyang et al., 2022; Longpre et al., 2023a, i.a.]. However, collecting such instruction data is costly and often suffers from limited diversity given that most human generations tend to be popular NLP tasks, falling short of covering a true variety of tasks and different ways to describe them. Continuing to improve the quality and coverage of instruction-tuned models necessitates the development of alternative approaches for supervising the instruction tuning process.

In this chapter, we introduce SELF-INSTRUCT, a semi-automated process for instruction-tuning a pre-trained LM using instructional signals from the model itself. The overall process is an iterative bootstrapping algorithm (see [Figure 3.1](#)), which starts off with a limited (e.g., 175 in our study) seed set of manually-written tasks that are used to guide the overall generation. In the first phase, the model is prompted to generate instructions for new tasks. This step leverages the existing collection of instructions to create more broad-coverage instructions that define (often new) tasks. Given the newly-generated set of instructions, the framework also creates input-output instances for them, which can be later used for supervising the instruction tuning. Finally, various heuristics are used to automatically filter low-quality or repeated instructions, before adding

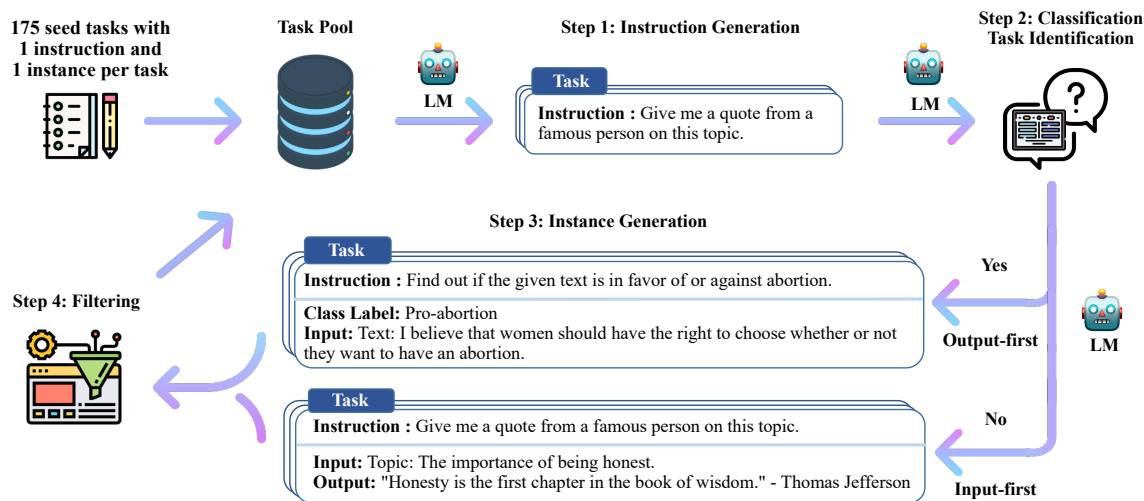


Figure 3.1: A high-level overview of SELF-INSTRUCT. The process starts with a small seed set of tasks as the task pool. Random tasks are sampled from the task pool, and used to prompt an off-the-shelf LM to generate both new instructions and corresponding instances, followed by filtering low-quality or similar generations, and then added back to the initial repository of tasks. The resulting data can be used for the instruction tuning of the language model itself later to follow instructions better. Tasks shown in the figure are generated by GPT3.

the remaining valid tasks to the task pool. This process can be repeated for many iterations until reaching a large number of tasks.

To evaluate SELF-INSTRUCT empirically, we run this framework on GPT3 [Brown et al., 2020a], which is a vanilla LM (§3.4). The iterative SELF-INSTRUCT process on this model leads to about 52k instructions, paired with about 82K instance inputs and target outputs. We observe that the resulting data provides a diverse range of creative tasks, as is demonstrated by examples in Figure 3.2. These generated tasks deviate from the distribution of typical NLP tasks, and also have fairly small overlap with the seed tasks (§3.4.2). On this resulting data, we build $GPT3_{\text{SELF-INST}}$ by finetuning GPT3 (i.e., the same model used for generating the instruction data). We evaluate $GPT3_{\text{SELF-INST}}$ in comparison to various other models on both typical NLP tasks included in SUPERNI [Wang et al., 2022b], and a set of new instructions that are created for novel usage of instruction-following models (§3.5). The results indicate that $GPT3_{\text{SELF-INST}}$ outperforms GPT3 (the original model) by a large margin (+33.1%) and nearly matches the performance of InstructGPT_{001} . Moreover, our human evaluation on the newly-created instruction set shows that $GPT3_{\text{SELF-INST}}$ demonstrates a broad range of instruction following ability, outperforming models trained on other publicly available

instruction datasets and leaving only a 5% gap behind InstructGPT₀₀₁¹.

In summary, our contributions are: (1) we introduce SELF-INSTRUCT, a method for inducing instruction following capabilities with minimal human-labeled data; (2) we demonstrate its effectiveness via extensive instruction-tuning experiments; and (3) we release a large synthetic dataset of 52K instructions and a set of manually-written novel tasks for building and evaluating future instruction-following models.

3.2 Background

Instruction-following LMs. A series of works have found evidence that vanilla LMs can be effective at following general language instructions if tuned with annotated “instructional” data—datasets containing language instructional commands and their desired outcomes based on human annotation [Weller et al., 2020; Mishra et al., 2022b; Wei et al., 2022a; Sanh et al., 2022, i.a.]. Additionally, they show a direct

correlation between the size and diversity of the “instructional” data and the generalizability of resulting models to unseen tasks [Wang et al., 2022b; Chung et al., 2022]. However, since these developments largely focus on existing NLP tasks and depend on human-annotated instructions, this poses a bottleneck for progress toward more generalizable models. Our work aims to move beyond classical NLP tasks and tackle the challenges of creating diverse instruction data by employing pretrained LMs. InstructGPT [Ouyang et al., 2022] shares a similar goal as ours in building more general-purpose LMs, and has demonstrated remarkable

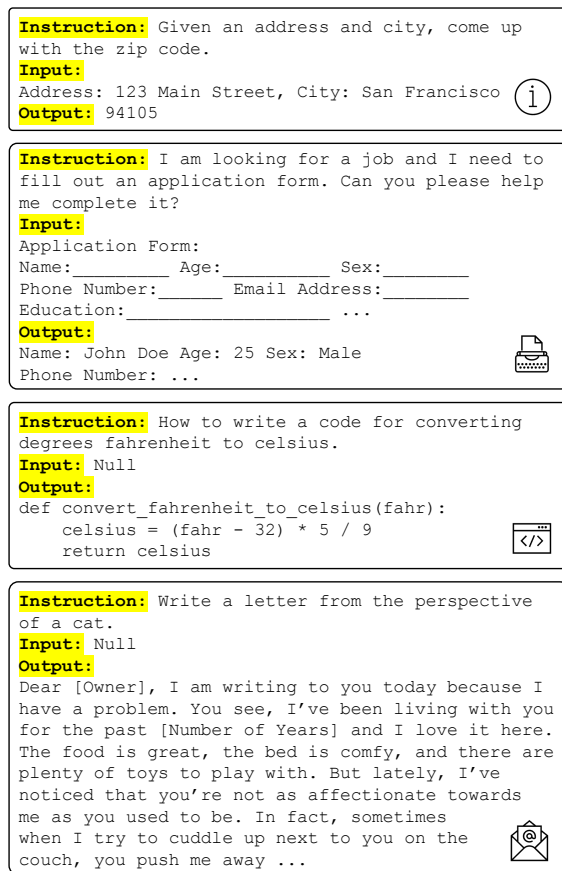


Figure 3.2: Selected tasks from the generated instruction data using vanilla GPT3. Some texts are reformatted for presentation. See Table B.7 for more examples.

¹Unless otherwise specified, our comparisons are with text-davinci-001 engine. We focus on this engine since it is the closest to our experimental setup: supervised finetuning with human demonstrations. The newer engines are more powerful, though use more data (e.g., code or latest user queries) or algorithms (e.g., PPO) that are difficult to compare with.

performance in following diverse user instructions. However, as a commercial system, their construction process still remains quite opaque. In particular, the role of *data* has remained understudied due to limited transparency and the private user data they used in their study. Addressing such challenges necessitates the creation of a large-scale, public dataset covering a broad range of tasks.

Language models for data generation and augmentation. A variety of works have proposed using LMs for data generation [Schick and Schütze, 2021; Wang et al., 2021; Liu et al., 2022; Meng et al., 2022] or augmentation [Feng et al., 2021; Yang et al., 2020; Mekala et al., 2022]. Our work differs from this line in that it is *not* specific to a particular task (say, QA or NLI). In contrast, a distinct motivation for SELF-INSTRUCT is to bootstrap new task definitions that may not have been defined before by NLP practitioners (though potentially still important for real users). In parallel with our work, Honovich et al. [2022a] also proposed to generate large-scale instruction data (so-called Unnatural Instructions) with GPT3 models. The major differences are that 1) they use tasks in SUPERNI as their seed tasks, resulting in a different distribution of generated tasks; 2) they employ InstructGPT₀₀₂ for generating the data, in which sense they are distilling knowledge from an already instruction-tuned model, while we solely rely on the vanilla LM; 3) the detailed generation pipeline and templates are different. Nevertheless, we believe that both efforts in expanding instruction data are complementary, and the community will benefit from these diverse datasets.

Instruction generation. A series of recent works [Zhou et al., 2022b; Ye et al., 2022; Singh et al., 2022; Honovich et al., 2022b] generate instructions of a task given a few examples. While SELF-INSTRUCT also involves instruction generation, a major difference in our case is it is task-agnostic; we generate new tasks (instructions along with instances) from scratch.

Model self-training. A typical self-training framework [He et al., 2019; Xie et al., 2020; Du et al., 2021; Amini et al., 2022; Huang et al., 2022] uses trained models to assign labels to unlabeled data and then leverages the pseudo-labeled data to improve the model. Zhou et al. [2022a] use multiple prompts to specify a single task and propose to regularize via prompt consistency, encouraging consistent predictions over the prompts. This allows either finetuning the model with extra unlabeled training data, or direct application at inference time. While SELF-INSTRUCT has similarities with the self-training literature, most self-training

methods assume a specific *target task* as well as *unlabeled examples* under it; in contrast, SELF-INSTRUCT produces a variety of tasks from scratch.

Knowledge distillation. Knowledge distillation [Hinton et al., 2015; Sanh et al., 2019; West et al., 2021; Magister et al., 2022] often involves the transfer of knowledge from larger models to smaller ones. SELF-INSTRUCT can also be viewed as a form of “knowledge distillation”, however, it differs from this line in the following ways: (1) the source and target of distillation are the same, i.e., a model’s knowledge is distilled to itself; (2) the content of distillation is in the form of an instruction task (i.e., instructions that define a task, and a set of examples that instantiate it).

Bootstrapping with limited resources. A series of recent works use language models to bootstrap some inferences using specialized methods. NPPrompt [Zhao et al., 2022] provides a method to generate predictions for semantic labels without any finetuning. It uses a model’s own embeddings to automatically find words relevant to the label of the data sample and hence reduces the dependency on manual mapping from model prediction to label (verbalizers). STAR [Zelikman et al., 2022] iteratively leverages a small number of rationale examples and a large dataset without rationales, to bootstrap a model’s ability to perform reasoning. Self-Correction [Welleck et al., 2022] decouples an imperfect base generator (model) from a separate corrector that learns to iteratively correct imperfect generations and demonstrates improvement over the base generator. Our work instead focuses on bootstrapping new tasks in the instruction paradigm.

Multi-modal instruction-following. Instruction-following models have also been of interest in the multi-modal learning literature [Fried et al., 2018; Shridhar et al., 2020; Min et al., 2022b; Weir et al., 2022]. SELF-INSTRUCT, as a general approach to expanding data, can potentially also be helpful in those settings, which we leave to future work.

3.3 Self-Instruct Pipeline

Annotating large-scale instruction data can be challenging for humans because it requires 1) creativity to come up with novel tasks and 2) expertise for writing the solutions to each task. Here, we detail our process for SELF-INSTRUCT, which refers to the pipeline of generating tasks with a *vanilla pretrained language*

model itself, filtering the generated data, and then conducting instruction tuning with this generated data in order to align the LM to follow instructions better. This pipeline is depicted in [Figure 3.1](#).

3.3.1 Defining Instruction Data

The instruction data we want to generate contains a set of instructions $\{I_t\}$, each of which defines a task t in natural language. Task t has $n_t \geq 1$ input-output instances $\{(X_{t,i}, Y_{t,i})\}_{i=1}^{n_t}$. A model M is expected to produce the output, given the task instruction and the corresponding input: $M(I_t, X_{t,i}) = Y_{t,i}$, for $i \in \{1, \dots, n_t\}$. Note that the instruction and instance input does not have a strict boundary in many cases. For example, “write an essay about school safety” can be a valid instruction that we expect models to respond to directly, while it can also be formulated as “write an essay about the following topic” as the instruction, and “school safety” as an instance input. To encourage the diversity of the data format, we allow such instructions that do not require additional input (i.e., X is empty).

3.3.2 Automatic Instruction Data Generation

Our pipeline for data generation consists of four steps: 1) generating task instructions, 2) determining if the instruction represents a classification task, 3) instance generation with either an input-first or output-first approach, and 4) filtering low-quality data.

Instruction Generation. At the first step, SELF-INSTRUCT generates new instructions from a small set of seed human-written instructions in a bootstrapping fashion. We initiate the task pool with 175 tasks (1 instruction and 1 instance for each task).² For every step, we sample 8 task instructions from this pool as in-context examples. Of the 8 instructions, 6 are from the human-written tasks, and 2 are from the model-generated tasks in previous steps to promote diversity. The prompting template is shown in [Table B.2](#).

Classification Task Identification. Because we need two different approaches for classification and non-classification tasks, we next identify whether the generated instruction represents a classification task or not.³ We prompt the LM in a few-shot way to determine this, using 12 classification instructions and 19

²These tasks were newly written by the authors and their labmates at UW, without reference to existing datasets or the test set used in this work. We provide more details about these tasks and analyze their similarity to the test tasks in [Appendix §B.1.1](#).

³More concretely, we regard tasks that have a small limited output label space as classification tasks.

non-classification instructions from the seed tasks. The prompting template is shown in [Table B.3](#).

Instance Generation. Given the instructions and their task type, we generate instances for each instruction independently. This is challenging because it requires the model to understand what the target task is, based on the instruction, figure out what additional input fields are needed and generate them, and finally complete the task by producing the output. We found that pretrained LMs can achieve this to a large extent when prompted with instruction-input-output in-context examples from other tasks. A natural way to do this is the **Input-first Approach**, where we can ask an LM to come up with the input fields first based on the instruction, and then produce the corresponding output. This generation order is similar to how models are used to respond to instruction and input, but here with in-context examples from other tasks. The prompting template is shown in [Table B.4](#).

However, we found that this approach can generate inputs biased toward one label, especially for classification tasks (e.g., for grammar error detection, it usually generates grammatical input). Therefore, we additionally propose an **Output-first Approach** for classification tasks, where we first generate the possible class labels, and then condition the input generation on each class label. The prompting template is shown in [Table B.5](#).⁴ We apply the output-first approach to the classification tasks identified in the former step, and the input-first approach to the remaining non-classification tasks.

Filtering and Postprocessing. To encourage diversity, a new instruction is added to the task pool only when its ROUGE-L similarity with any existing instruction is less than 0.7. We also exclude instructions that contain some specific keywords (e.g., image, picture, graph) that usually can not be processed by LMs. When generating new instances for each instruction, we filter out instances that are exactly the same or those with the same input but different outputs. Invalid generations are identified and filtered out based on heuristics (e.g., instruction is too long or too short, instance output is a repetition of the input).

⁴In this work, we use a fixed set of seed tasks for prompting the instance generation, and thus only generate a small number of instances per task in one round. Future work can use randomly sampled tasks to prompt the model to generate a larger number of instances in multiple rounds.

3.4 Synthesized Data from GPT3

In this section, we apply our method for inducing instruction data to GPT3 as a case study. We use the largest GPT3 LM (“davinci” engine) accessed through the OpenAI API.⁵ The parameters for making queries are described in Appendix B.1.2. Here we present an overview of the generated data.

3.4.1 Statistics

Table 3.1 describes the basic statistics of the generated data. We generate a total of over 52K instructions and more than 82K instances corresponding to these instructions after filtering.

statistic	
# of instructions	52,445
- # of classification instructions	11,584
- # of non-classification instructions	40,861
# of instances	82,439
- # of instances with empty input	35,878
ave. instruction length (in words)	15.9
ave. non-empty input length (in words)	12.7
ave. output length (in words)	18.9

Table 3.1: Statistics of the generated data by applying SELF-INSTRUCT to GPT3.

3.4.2 Diversity

To study what types of instructions are generated and how diverse they are, we identify the verb-noun structure in the generated instructions. We use the Berkeley Neural Parser⁶ [Kitaev and Klein, 2018; Kitaev et al., 2019] to parse the instructions and then extract the verb that is closest to the root as well as its first direct noun object. 26,559 out of the 52,445 instructions contain such structure; other instructions usually contain more complex clauses (e.g., “Classify whether this tweet contains political content or not.”) or are framed as questions (e.g., “Which of these statements are true?”). We plot the top 20 most common root verbs and their top 4 direct noun objects in Figure 3.3, which account for 14% of the entire set. Overall, we see quite diverse intents and textual formats in these instructions.

⁵<https://openai.com/api/>

⁶<https://parser.kitaev.io/>

We further study how the generated instructions differ from the seed instructions used to prompt the generation. For each generated instruction, we compute its highest ROUGE-L overlap with the 175 seed instructions. We plot the distribution of these ROUGE-L scores in [Figure 3.4](#). The results indicate a decent number of new instructions were generated, which do not have much overlap with the seeds. We also demonstrate diversity in the length of the instructions, instance inputs, and instance outputs in [Figure 3.5](#).

3.4.3 Quality

So far, we have shown the quantity and diversity of the generated data, but its quality remains uncertain. To investigate this, we randomly sample 200 instructions and randomly select 1 instance per instruction. We asked an expert annotator (author of this work) to label whether each instance is correct or not, in terms of the instruction, the instance input, and the instance output. Evaluation results in [Table 3.2](#) show that most of the generated instructions are meaningful, while the generated instances may contain more noise (to a reasonable extent). However, we found that even though the generations may contain errors, most of them are still in the correct format or partially correct, which can provide useful guidance for training models to follow instructions. We listed a number of good examples and bad examples in [Table B.7](#) and [B.8](#), respectively.

Quality Review Question	Yes %
Does the instruction describe a valid task?	92%
Is the input appropriate for the instruction?	79%
Is the output a correct and acceptable response to the instruction and input?	58%
All fields are valid	54%

Table 3.2: Data quality review for the instruction, input, and output of the generated data.

3.5 Results

We conduct experiments to measure and compare the performance of models under various instruction tuning setups. We first describe our models and other baselines, followed by our experiments.

3.5.1 GPT3_{SELF-INST}: a self-instructed GPT3 model

Given the instruction-generated instruction data, we conduct instruction tuning with the GPT3 model itself (“davinci” engine). As described in §3.3.3, we use various templates to concatenate the instruction and input, and train the model to generate the output. This finetuning is done through the OpenAI finetuning API.⁷ We use the default hyper-parameters, except that we set the prompt loss weight to 0, and we train the model for 2 epochs. We refer the reader to Appendix B.1.3 for additional finetuning details. The resulting model is denoted by GPT3_{SELF-INST}.

3.5.2 Baselines

Off-the-shelf LMs. We evaluate T5-LM [Lester et al., 2021; Raffel et al., 2020] and GPT3 [Brown et al., 2020a] as the vanilla LM baselines (only pretraining, no additional finetuning). These baselines will indicate the extent to which off-the-shelf LMs are capable of following instructions naturally immediately after pretraining.

Publicly available instruction-tuned models. T0 and T k -INSTRUCT are two instruction-tuned models proposed in Sanh et al. [2022] and Wang et al. [2022b], respectively, and are demonstrated to be able to follow instructions for many NLP tasks. Both of these models are finetuned from the T5 [Raffel et al., 2020] checkpoints and are publicly available.⁸ For both of these models, we use their largest version with 11B parameters.

Instruction-tuned GPT3 models. We evaluate InstructGPT [Ouyang et al., 2022], which is developed by OpenAI based on GPT3 to follow human instructions better and has been found by the community to have impressive zero-shot abilities. There are various generations of these models, where newer ones use more expansive data or algorithmic novelties.⁹ For our SUPERNI experiments in §3.5.3, we only compare with their text-davinci-001 engine, because their newer engines are trained with the latest user data and are likely to have already seen the SUPERNI test set. For our human evaluation on newly written instructions, we include their 001, 002 and 003 engines for completeness.

⁷See [OpenAI’s documentation on finetuning](#).

⁸T0 is available at [here](#) and T k -INSTRUCT is [here](#).

⁹See [OpenAI’s documentation on their models](#).

Model	# Params	ROUGE-L
Vanilla LMs		
T5-LM	11 B	25.7
GPT-3	175 B	6.8
Instruction-tuned without Super-NI		
T0	11 B	33.1
GPT-3 + T0 training	175 B	37.9
GPT-3 + Self-Instruct (ours)	175 B	39.9
InstructGPT-001	175 B	40.8
Instruction-tuned with Super-NI		
TK-Instruct	11 B	46.0
GPT-3 + Super-NI training	175 B	49.5
GPT-3 + Self-Instruct + Super-NI (ours)	175 B	51.6

Table 3.3: Evaluation results on *unseen* tasks from Super-NI.

Additionally, to compare SELF-INSTRUCT training with other publicly available instruction tuning data, we further finetune GPT3 model with data from PROMPTSOURCE and SUPERNI, which are used to train the T0 and Tk-INSTRUCT models. We call them T0 training and SUPERNI training for short, respectively. To save the training budget, we sampled 50K instances (but covering all their instructions) for each dataset, which has a comparable size to the instruction data we generated. Based on the findings from Wang et al. [2022b] and our early experiments, reducing the number of instances per task does not degrade the model’s generalization performance to unseen tasks.

3.5.3 Zero-Shot Generalization on SUPERNI benchmark

We first evaluate the models’ ability to follow instructions on typical NLP tasks in a zero-shot fashion. We use the evaluation set of SUPERNI, which consists of 119 tasks with 100 instances in each task. In this work, we mainly focus on the zero-shot setup, i.e., the model is prompted with the definition of the tasks only, without in-context demonstration examples. For all our requests to the GPT3 variants, we use the deterministic generation mode (temperature as 0 and no nucleus sampling) without specific stop sequences.

Results. We make the following observations from the results in Table 3.3. SELF-INSTRUCT boosts the instruction-following ability of GPT3 by a large margin. The vanilla GPT3 model basically cannot follow

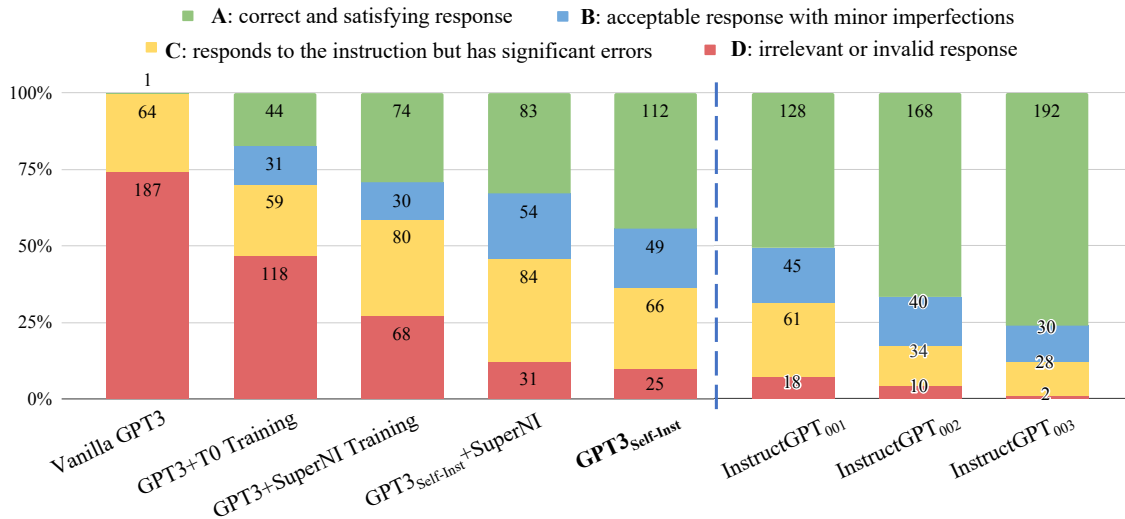


Figure 3.6: Performance of GPT3 model and its instruction-tuned variants, evaluated by human experts on our 252 user-oriented instructions (§3.5.4). Human evaluators are instructed to rate the models’ responses into four levels. The results indicate that GPT3_{SELF-INST} outperforms all the other GPT3 variants trained on publicly available instruction datasets. Additionally, GPT3_{SELF-INST} scores nearly as good as InstructGPT₀₀₁ (cf. Figure 1).

human instructions at all. Upon manual analysis, we find that it usually generates irrelevant and repetitive text, and does not know when to stop generation. Compared with other models that are not specifically trained for SUPERNI, GPT3_{SELF-INST} achieves better performance than T0 or the GPT3 finetuned on the T0 training set, which takes tremendous human labeling efforts. Notably, GPT3_{SELF-INST} also nearly matches the performance of InstructGPT₀₀₁, which is trained with private user data and human-annotated labels.

Models trained on the SUPERNI training set still achieve better performance on its evaluation set, which we attribute to the similar instruction style and formatting. However, we show that SELF-INSTRUCT still brings in additional gains when combined with the SUPERNI training set, proving its value as complementary data.

3.5.4 Generalization to User-oriented Instructions on Novel Tasks

Despite the comprehensiveness of SUPERNI in collecting existing NLP tasks, most of these NLP tasks were proposed for research purposes and skewed toward classification. To better access the practical value of instruction-following models, a subset of the authors curate a new set of instructions motivated by user-oriented applications. We first brainstorm various domains where large LMs may be useful (e.g., email writing, social media, productivity tools, entertainment, programming), then craft instructions related to

each domain along with an input-output instance (again, input is optional). We aim to diversify the styles and formats of these tasks (e.g., instructions may be long or short; input/output may take the form of bullet points, tables, codes, equations, etc.). In total, we create 252 instructions with 1 instance per instruction. We believe it can serve as a testbed for evaluating how instruction-based models handle diverse and unfamiliar instructions. [Table B.6](#) presents a small portion of them. The entire set is available in our GitHub repository. We analyze the overlap between this set and the seed instructions in [§B.1.1](#).

Human evaluation setup. Evaluating models’ performance on this evaluation set of diverse tasks is extremely challenging because different tasks require different expertise. Indeed, many of these tasks cannot be measured by automatic metrics or even be judged by normal crowdworkers (e.g., writing a program or converting first-order logic into natural language). To get a more faithful evaluation, we asked the authors of the instructions to judge model predictions. Details on how we set up this human evaluation are described in [Appendix B.2](#). The evaluators were asked to rate the output based on whether it accurately and effectively completes the task. We implemented a four-level rating system for categorizing the quality of models’ outputs:

- RATING-A: The response is valid and satisfying.
- RATING-B: The response is acceptable but has minor errors or imperfections.
- RATING-C: The response is relevant and responds to the instruction, but it has significant errors in the content. For example, GPT3 might generate a valid output first, but continue to generate other irrelevant things.
- RATING-D: The response is irrelevant or completely invalid.

Results. [Figure 3.6](#) shows the performance of GPT3 model and its instruction-tuned counterparts on this newly written instruction set (w. inter-rater agreement $\kappa = 0.57$ on the 4-class categorical scale, see [Appendix B.2](#) for details). As anticipated, the vanilla GPT3 LM is largely unable to respond to instructions, and all instruction-tuned models demonstrate comparatively higher performance. Nonetheless, GPT3_{SELF-INST} (i.e., GPT3 model finetuned with SELF-INSTRUCT) outperforms those counterparts trained on T0 or SUPERNI

data by a large margin, demonstrating the value of the generated data despite the noise. Compared with InstructGPT₀₀₁, GPT3_{SELF-INST} is quite close in performance—if we count acceptable responses with minor imperfections (RATING-B) as valid, GPT3_{SELF-INST} is only 5% behind InstructGPT₀₀₁. Lastly, our evaluation confirms the impressive instruction-following ability of InstructGPT₀₀₂ and InstructGPT₀₀₃. Although there are many factors behind this success, we conjecture that future work can largely benefit from improving the quality of our generated data by using human annotators or training a reward model to select better generations, similar to the algorithm used by Ouyang et al. [2022].

3.5.5 Effect of Data Size and Quality

Data size. SELF-INSTRUCT provides a way to grow instruction data at a low cost with almost no human labeling; could more of this generated data lead to better instruction-following ability? We analyze the size of generated data by subsampling different numbers of instructions from the generated dataset, finetuning GPT3 on the sampled subsets, and evaluating how the resulting models perform on the 252 user-oriented instruction set. We conduct the same human evaluation as in §3.5.4. Figure 3.7 presents the performance of GPT3_{SELF-INST} models finetuned with different sizes of generated data. Overall, we see consistent improvement as we grow the data size. However, this improvement almost plateaus after 16K. This is in line with the data scaling experiments in the previous chapter

(§2.6). Interestingly, when evaluating on SUPERNI we found the model’s performance gain plateaus earlier at around hundreds of instructions. This may be due to the fact that the newly generated data is distinct from typical NLP tasks in SUPERNI, indicating that future research may benefit from using a combination of different instruction data for better performance on various types of tasks.

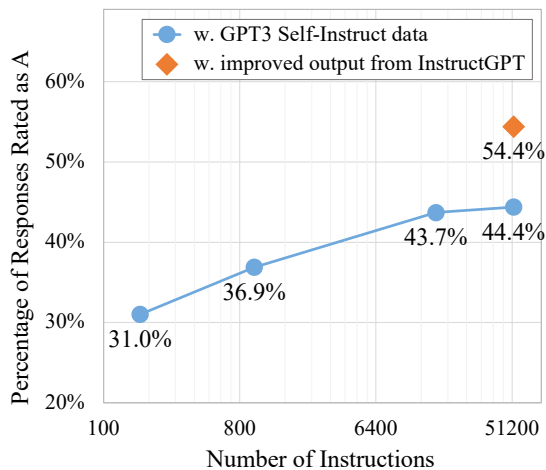


Figure 3.7: Human evaluation performance of GPT3_{SELF-INST} models tuned with different sizes of instructions. x -axis is in log scale. The smallest size is 175, where only the seed tasks are used for instruction tuning. We also evaluate whether improving the data quality further improves the performance by distilling the outputs from InstructGPT₀₀₃. We see consistent improvement from using larger data with better quality.

Data quality. Another direction to improve the model’s performance is to take our generated data and get better supervision (with less noise). We explore this idea by using InstructGPT₀₀₃ (the best available general-purpose model) to regenerate the output field of all our instances given the instruction and input. We then use this improved version of our data to finetune GPT3. As is shown in [Figure 3.7](#), the resulting model outperforms the counterpart trained with the original data by 10%, which suggests big room for future work on using our generation pipeline to get initial data and then improving the data quality with human experts or distillation from better models.

3.6 Discussion & Future work

Here, we discuss some limitations of this work to inspire future research in this direction.

Tail phenomena. SELF-INSTRUCT depends on LMs, and it will inherit all the limitations that carry over with LMs. As recent studies have shown [[Razeghi et al., 2022](#); [Kandpal et al., 2022](#)], *tail phenomena* pose a serious challenge to the success of LMs. In other words, LMs’ largest gains correspond to the frequent uses of languages (head of the language use distribution), and there might be minimal gains in the low-frequency contexts. Similarly, in the context of this work, it would not be surprising if the majority of the gains by SELF-INSTRUCT are skewed toward tasks or instructions that present more frequently in the pretraining corpus. As a consequence, the approach might show brittleness with respect to uncommon and creative instructions.

Dependence on large models. Because of SELF-INSTRUCT’s dependence on the inductive biases extracted from LMs, it might work best for larger models. If true, this may create barriers to access for those who may not have large computing resources. We hope future studies will carefully study the gains as a function of model size or various other parameters. It is worthwhile to note that instruction-tuning with human annotation also suffers from a similar limitation: gains of instruction-tuning are higher for larger models [[Wei et al., 2022a](#)].

Reinforcing LM biases. A point of concern for the authors is the unintended consequences of this iterative algorithm, such as the amplification of problematic social biases (stereotypes or slurs about gender, race, etc.).

Relatedly, one observed challenge in this process is the algorithm’s difficulty in producing balanced labels, which reflected models’ prior biases. We hope future work will lead to better understanding of the pros and cons of the approach.

3.7 Summary

We introduce SELF-INSTRUCT, a method to improve the instruction-following ability of LMs via their own generation of instruction data. On experimenting with vanilla GPT3, we automatically construct a large-scale dataset of 52K instructions for diverse tasks, and finetuning GPT3 on this data leads to a 33% absolute improvement on SUPERNI over the original GPT3. Furthermore, we curate a set of expert-written instructions for novel tasks. Human evaluation on this set shows that tuning GPT3 with SELF-INSTRUCT outperforms using existing public instruction datasets by a large margin and performs closely to InstructGPT₀₀₁. We hope SELF-INSTRUCT can serve as the first step to align pretrained LMs to follow human instructions, and future work can build on top of this data to improve instruction-following models.

Chapter 4

Routing Annotations between Human and AI

4.1 Overview

Reinforcement learning from human feedback [Christiano et al., 2017] has been integral to the alignment of large language models (LMs) with human objectives and values [Ouyang et al., 2022; Bai et al., 2022a, *inter alia*]. Central to this process are preference datasets, i.e., input instances to language models paired with candidate model outputs and human judgment annotations indicating the preferred output. Collecting preference data involves several key design decisions, and one important consideration is determining the source of preference annotations [Kirk et al., 2023, 2024]. This choice impacts not only the cost and effort required to procure these annotations, but also the performance of models trained on them.

There are two major approaches to obtain preference annotations. One approach is to solicit **preferences directly from humans**. Although this setup leads to generally high-quality data [Wang et al., 2024c], the annotation process itself is expensive and time-consuming. Moreover, human annotators can make mistakes, especially when faced with complex examples or when the content extends beyond their expertise [Jiang and de Marneffe, 2022; Sandri et al., 2023]. In contrast, preference annotations can also be obtained indirectly from humans by querying an off-the-shelf LM trained on human preferences [Bai et al., 2022b; Lee et al., 2023; Cui et al., 2023], leading to a set of **synthetic preferences**. Although this approach is more scalable, LMs do not always reflect the nuances of human annotators and can be prone to certain biases or errors in judgment [Singhal et al., 2023; Wang et al., 2024a]. Hence, we posit that obtaining high-quality and

cost-efficient preference data involves finding the right combination of direct human and synthetic preferences from LMs.

We present **HYPER**, a **Hybrid Preference routER** that allocates preference instances to human or LM annotators, resulting in a set of **hybrid annotations** (§4.3). The crux of our approach is to identify specific instances that will benefit from direct human annotations, while the rest are routed to the LM. We ground this decision in the performance of RMs trained on the resulting preference datasets, measured by RewardBench [Lambert et al., 2024b]. HYPER consists of a **performance prediction model** (PPM, §4.3.2) and a **routing strategy** (§4.3.3) as illustrated in Figure 4.1. The PPM learns to predict the performance of a model trained on a preference dataset based on the statistics of the subset being routed to human annotators. We then use the PPM to predict the performance of arbitrary simulated hybrid datasets in order to recommend the potentially best-performing one. To put HYPER into practice, we construct **MULTIPREF**, a preference dataset containing 10k instances paired with both human and LM preference annotations that follow the same carefully designed annotation guidelines (§4.4). Then, we train the PPM on this dataset and use the routing strategy to obtain hybrid annotations from either LMs or humans.

Our experiments show that hybrid annotations constructed from HYPER’s predictions result in better RMs than those trained (a) entirely on direct human preferences, (b) entirely on synthetic preferences, and (c) a random combination of direct human and synthetic preferences given the same human annotation budget (§4.5), supporting our hypothesis that there exist optimal combinations of annotations that are not exclusively direct human or synthetic. Our results generalize across other existing preference datasets (§4.5.2), base models (§4.5.3), and common LM benchmarks through best-of-N reranking (§4.5.4). The resulting hybrid preference datasets outperform the corresponding original ones by a large margin, with 7–13% (absolute) improvement on RewardBench and up to 3% (absolute) improvement on downstream evaluations on average, demonstrating HYPER’s generalization capabilities. We then present an analysis of factors that render a preference instance to benefit from direct human annotations (§4.6).

Overall, we hope that this work provides a more cost-effective approach to preference data collection while providing actionable, data-centric insights on preference learning.

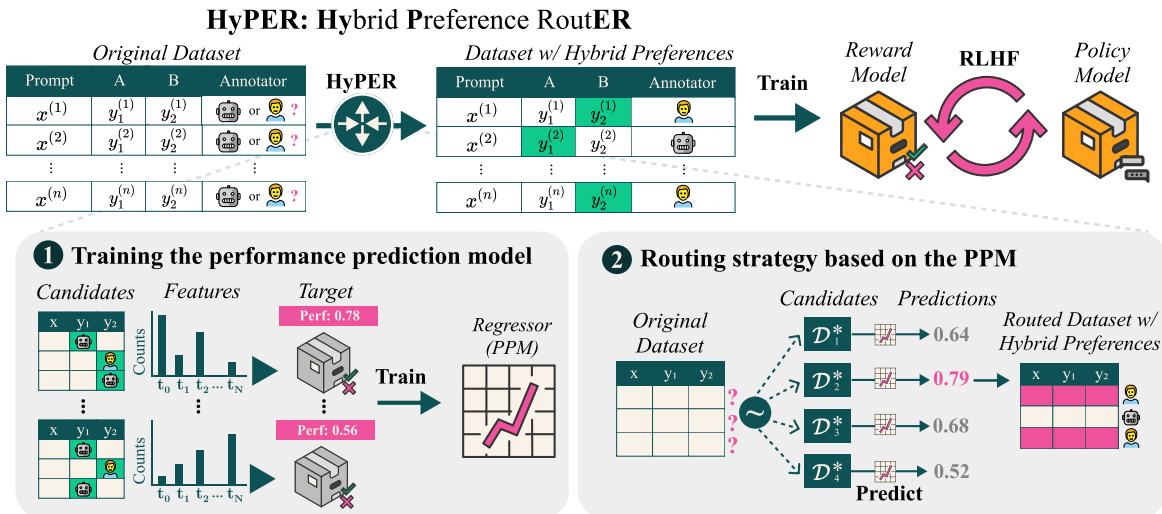


Figure 4.1: Overview of HyPER. Our proposed method consists of a performance prediction model (PPM) and a routing strategy based on that model. We train the PPM to predict the performance of a dataset based on the statistics of the subset routed to human annotators. Then, we use the PPM to score many simulations of candidate datasets, and recommend the potentially best-performing routing configuration.

4.2 Background

Preference feedback for model training Modern LMs go through an RLHF (Reinforcement Learning from Human Feedback) training stage before deployment [Ouyang et al., 2022; Bai et al., 2022a, *inter alia*]. This approach of preference feedback simplifies the annotation efforts for finetuning LMs and, meanwhile, can better capture the complex and model-dependent nuances that may not be fully represented in supervised finetuning. Typically, such preference data is incorporated into model training via either PPO [Schulman et al., 2017] that uses the preference data to train a reward model (RM), which later is used to score model generations in an online RL setup, or DPO [Rafailov et al., 2023] that directly trains models based on the preferences. In this work, we mainly focus on the RM part by directly evaluating RMs on RewardBench [Lambert et al., 2024b] and Best-of-N reranking performance [Iverson et al., 2024].

Data mixing and selection in LM training. Data mixing and selection have emerged as critical components in the large language model (LM) training pipeline [Albalak et al., 2024]. Various studies have addressed these challenges in different stages of the LM training process, particularly in pretraining [Xie et al., 2024; Liu et al., 2024, *inter alia*] and supervised fine-tuning [Wang et al., 2023b; Lu et al., 2023; Xia et al., 2024,

inter alia). A notable contribution by Ivison et al. [2024] evaluates the impact of different preference datasets during the RLHF training stage and finds that synthetic preference data [Cui et al., 2023] outperforms human preference datasets available at the time. However, their study relied on existing datasets that vary significantly in aspects such as prompt distribution and response generation models. Our work, HYPER, is a novel routing framework aimed at optimizing in the preference label space, featuring an automated algorithm to select the appropriate annotation source, utilizing human input only when necessary. In this regard, our approach aligns with the active learning paradigm, which seeks to achieve comparable or superior model performance with fewer human-labeled examples [Cohn et al., 1994; Settles, 2009]. In relation to this paradigm, another framework called *CoAnnotating* [Li et al., 2023a], uses uncertainty measurements such as entropy and an LM’s self-evaluation in order to decide whether an annotation instance will be allocated to humans or LMs. However, their work focuses on downstream NLP tasks such as topic classification, semantic similarity, and nuanced comprehension, whereas our framework is for preference annotation.

Performance Prediction HYPER relies on a performance prediction model (PPM) to predict the performance metric given a dataset. This problem has been studied before based on various factors [Birch et al., 2008; Xia et al., 2020; Ye et al., 2021b]. Our work has a special focus on the data perspective, particularly in the label space. Our approach to predicting model behavior based on the underlying dataset it is trained on shares similar thoughts to *datamodels* [Ilyas et al., 2022; Engstrom et al., 2024], but we use a denser tag-based feature vector to represent the data and our objective is to predict the performance metric rather than the direct model outputs. Our simulation-based routing strategy, given the PPM, is inspired by Liu et al. [2024], which studies domain mixing in the pretraining stage.

4.3 Learning to Route Preference Annotations

4.3.1 Problem Formulation

We first formulate the preference routing problem. Let $\mathcal{D} = \{\langle x^{(i)}, y_1^{(i)}, y_2^{(i)} \rangle\}_{i=1}^n$ be a dataset of n unlabeled preference instances containing prompts x and pairwise responses y_1 and y_2 , where each instance can be assigned a label from either of the two sources: one provided by a human annotator, or one generated by an LM. We introduce a binary decision variable $z_i \in \{0, 1\}$ for each instance, where $z_i = 0$ corresponds to

selecting the human-provided label and $z_i = 1$ corresponds to selecting the LM-generated label. Note that z_i denotes the source of the labels, and not the identity of the labels—when the humans and the LM agree, the chosen label is the same irrespective of z_i .

The goal for routing is to optimize the selection of binary decision variables z_i for the dataset in order to maximize a performance metric. This optimization problem can be expressed as:

$$\max_{z \in \{0,1\}^n} \text{PERF}(R(\mathcal{D}(z))), \quad (4.1)$$

where $\text{PERF}(R(\mathcal{D}(z)))$ is the performance of a reward model R trained on dataset $\mathcal{D}(z)$. Here, $z = \{z_1, z_2, \dots, z_n\}$ is the *routing configuration*, representing the vector of binary label choices for all instances. Maximizing Equation 4.1 is difficult as there is no closed-form solution. In addition, finding the best routing configuration is computationally heavy, as brute force search would entail training and evaluating a reward model for 2^n configurations. Instead, we convert the problem into a learning objective, where we train a model to predict the reward performance of a given routing configuration. We construct *candidate* labeled datasets $\hat{\mathcal{D}}(z)$ with different routing configurations z which we use to train reward models, denoted as $\hat{R}(\hat{\mathcal{D}}(z))$.¹ We use these candidates to train a **performance prediction model** that approximates $\text{PERF}(\hat{R}(*))$ (§4.3.2). After training the model, we use a simulation-based **routing strategy** that aims to find the optimal z to maximize the predicted performance (§4.3.3).

4.3.2 Performance Prediction Model (PPM)

PPM is a regression model that provides an estimate of the performance of a reward model trained on a candidate preference dataset $\hat{\mathcal{D}}$. The PPM takes as input a feature vector representing the routing configuration of $\hat{\mathcal{D}}$ and outputs a scalar value as the predicted performance. Training the PPM requires a seed preference dataset \mathcal{D} with both human and LM labels to build multiple samples of candidate datasets $\{\hat{\mathcal{D}}_i\}$ with different routing configurations and their evaluation performance.

Step 1: Defining a Feature Vector. Instead of directly operating on individual preference instances, we define a feature space for the PPM so that we can make routing decisions about groups of instances that share

¹For the rest of this paper, we will ignore the z variable for simplicity and denote the candidate labeled dataset as $\hat{\mathcal{D}}$.

Algorithm 1 Generating a candidate dataset $\hat{\mathcal{D}}$

Require: Unrouted dataset $\mathcal{D} = \{d_1, d_2, \dots, d_N\}$, Mapping between tags t and instances with that tag, $M = \{t_i \mapsto \{d_j \in \mathcal{D} \mid d_j \text{ has tag } t_i\} \mid i = 1, 2, \dots, N\}$

- 1: Budget $b \sim \text{Uniform}(1, |\mathcal{D}| - 1)$ ▷ Sample a random budget
- 2: $S_{\text{human}} \leftarrow \{\}$ ▷ Initialize subset that will use human annotations
- 3: $M \leftarrow \text{SHUFFLE}(M)$ ▷ Shuffle order of feats.
- 4: **while** $|S_{\text{human}}| < b$ **do**
- 5: **for** m in M **do**
- 6: $S_{\text{human}} \leftarrow m$ ▷ Add instances associated with tag m to S_{human}
- 7: **end for**
- 8: **end while**
- 9: $z \leftarrow \{0 \text{ if } d_i \in S_{\text{human}} \text{ else } 1 \mid d_i \in \mathcal{D}\}$
- 10: $\hat{\mathcal{D}} \leftarrow \mathcal{D}(z)$
- 11: **return** $\hat{\mathcal{D}}$

features, allowing our routing procedure to generalize to other datasets where these features might be present. We construct a feature space of **tags** T - textual and descriptive features of an instance’s prompt-response triples:

- **Textual tags** characterize textual information such as the cosine similarity of the encoded representation² of the responses y_1 and y_2 , the length of the prompt x , or the token length difference between two responses. We discretize the textual tags to convert them into categorical bins.
- **Descriptive tags** include metadata about the prompt or instruction such as the *subject of expertise* needed to answer the prompt, or the *complexity of user intent* in the prompt based on the number of goals or requirements among many others. We obtain these descriptors from a multilabel classifier trained on a human-validated dataset of instructions and their corresponding tags (see Appendix C.4.1 for more details).

We then represent each candidate dataset as a vector $v = \{C_{t_j, \text{human}} \mid t_j \in T\}$, where $C_{t_j, \text{human}}$ denotes the count of instances routed to human annotations with the j^{th} tag. The full list of tags can be found in Appendix C.4.

Step 2: Constructing Candidate Datasets and Measuring their Performance. We generate candidate datasets $\{\hat{\mathcal{D}}_i\}$ from the original unrouted dataset \mathcal{D} by sampling different routing configurations z as shown in

²We use the all-distilroberta-v1 embedding model from sentence-transformers [Reimers and Gurevych, 2019].

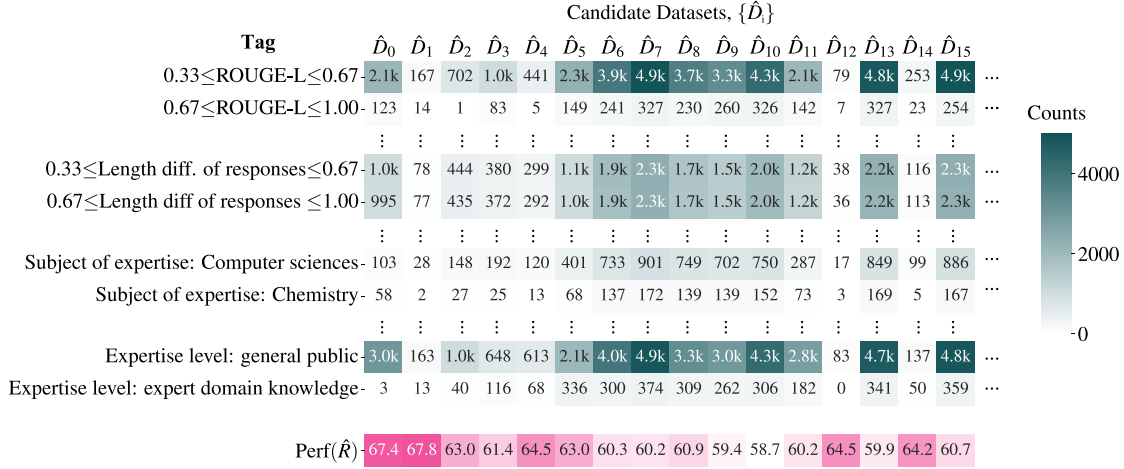


Figure 4.2: Feature representation of candidate datasets and their actual reward modeling performance as the training data for PPM. We use the count of instances that belong to the human annotation subset S_{human} as the feature value for each tag, and the RewardBench overall accuracy as the target. This heatmap shows the features derived from MULTIPREF.

Algorithm 1. At a high level, this algorithm generates a candidate dataset by randomly selecting the number of instances b to be annotated by humans or LM. Then, it iteratively adds instances associated with each tag to a subset until b is met, and assigns binary labels based on the subset to create the candidates. Some tags might get ignored once the number of instances reach b , so we shuffle their order to ensure that the tags are well represented between candidates.

We also include candidates where all preference labels are from humans ($|S_{\text{human}}| = |D|$) and all labels are from LMs ($|S_{\text{human}}| = 0$). Our sampling algorithm attempts to cover many human annotation budgets and different types of instances assigned to them. For each candidate dataset, we train a reward model \hat{R} and evaluate its performance $\text{PERF}(\hat{R})$ on an evaluation metric, in this case, RewardBench. This process leads to a PPM training dataset with the tag counts as features and the RM performance as the target as shown in Figure 4.2.

Step 3: Training the Performance Prediction Model. We fit a regression model to predict the RewardBench performance of a candidate dataset. We use the feature vector v as the features and the reward model performance on RewardBench $\text{PERF}(\hat{R})$ as the target. In practice, we collected 200 candidates \hat{D} and their performance from MULTIPREF for training the PPM.

4.3.3 Routing Strategy based on the PPM

Given a preference dataset \mathcal{D} , we also simulate candidates $\{\hat{\mathcal{D}}(z)\}$ using Algorithm 1 and predict their performance using the PPM from the previous stage (§4.3.2). We can simulate candidates with either a fixed human annotation budget, which is common in practice, or a range of random budgets to identify the optimal hybrid mix. Since the PPM estimates the expected performance of any $\hat{\mathcal{D}}_i$, we can simulate a large number of candidates and estimate their performance without training any RM.

For inference, our goal is to find the best routing configuration $z^* = \{z_1, z_2, \dots, z_n\}$ that will maximize RM performance $\text{PERF}(\hat{\mathcal{D}}(z^*))$. This configuration specifies which preference instances should be routed to humans or LMs that will result in the highest RewardBench score. To obtain z^* , we take the candidate with the highest predicted RM performance and use its configuration z for routing. For each preference instance d_i in \mathcal{D} , we take the decision z_i and route the instance to humans if $z_i = 0$ and to LMs if $z_i = 1$. In practice, we generate 500 candidates from which we select the best routing configuration.

4.3.4 Routing Strategy for a Single Instance

In order to route a single instance, we compute its *expected performance gain*. We define gain by measuring the improvement in RM performance if a particular instance is routed to humans for annotation. We calculate it by computing the difference between a (1) routing configuration where a specific instance is routed to human annotators and a (2) routing configuration where no instances are routed to human annotators (i.e., 100% synthetic annotations): $\Delta = \text{PPM}(v_n) - \text{PPM}(v_0)$. We then route a preference instance to human annotators if $\Delta > 0$ and to LMs otherwise.

4.4 MULTIPREF: A New Preference Dataset

We introduce MULTIPREF, a new preference dataset containing 10,461 instances with both human and GPT-4 annotations. We use MULTIPREF to train HYPER’s PPM. We collect prompts from datasets such as ShareGPT [Chiang et al., 2023], WildChat [Zhao et al., 2024], HH-RLHF [Bai et al., 2022a], and ChatArena [Chiang et al., 2024]. Then, we generate model responses using a variety of models, including Llama-2-Chat 70B [Touvron et al., 2023a], Llama-3-Instruct 70B [Dubey et al., 2024], TULU-2 7B and 70B [Iverson et al.,

Dataset statistics	
# unique prompts	5,323
# models for generation	6
# model pairs	21
# comparisons	10,461
# annotations	41,844
# annotation per instance	4
Annotator statistics	
Total # of crowdworkers	289
Avg. qualification test pass rate	34.8%

Table 4.1: MULTIPREF dataset statistics.

2023], GPT-3.5, and GPT-4 [Achiam et al., 2023].³

MULTIPREF is then annotated carefully to control for annotation quality, while working with crowdworkers on a fair wage (\$15-\$20 USD per hour based on expertise-level). We recruit annotators from Prolific,⁴ a crowdsourcing platform. We screened workers using a qualification test that filtered out 65% of the initial workers. Prolific implements various checks to avoid annotators using bots during the annotation. Each instance in MULTIPREF is annotated by four (4) crowdworkers. We aggregate these labels via majority vote to mitigate noise in annotation. We also collect LM annotations using GPT4 and include in its prompt the same guidelines we presented to human annotators. Since we allow ties during annotation, we filter instances that are labeled as a “Tie” by either human or GPT4, ending up with 7,531 non-tie preference instances that can be used for model training. Appendix C.1 shows additional information on the data collection process. Table 4.1 summarizes dataset statistics of MULTIPREF.

4.5 Experiments

We first intrinsically evaluate how well the PPM fits on a domain it was trained on (§4.5.1), then we assess how well the same PPM generalizes to other preference datasets (§4.5.2) and models (§4.5.3) on the same target evaluation metric (RewardBench). Finally, we test how well HYPHER generalizes to other LM benchmarks on various preference datasets (§4.5.4).

³We use model versions gpt-3.5-turbo-0125 and gpt-4-turbo-2024-04-09 for GPT-3.5 and GPT-4, respectively.

⁴<https://www.prolific.com/>

Model type	Spearman $\rho \uparrow$	RMSE \downarrow
Linear	0.408 ± 0.056	0.311 ± 0.044
LightGBM	0.127 ± 0.009	0.425 ± 0.010
Quadratic	0.610 ± 0.042	0.266 ± 0.054

Table 4.2: Using 10-fold cross validation on 250 candidate datasets, we report the average Spearman ρ of predicted vs actual ranks and RMSE of predicted vs actual RewardBench performance.

4.5.1 Performance Prediction Model Details

Testing the PPM’s fit. In order to test whether the PPM can accurately predict the performance of a preference dataset on RewardBench, we perform 10-fold cross-validation on 250 candidates from MULTIPREF (225 instances for training and 25 instances for validation). For each fold, we train a regressor to predict the performance of the held-out set and evaluate it with the actual RewardBench score. We evaluate the regression models using root-mean-square error (RMSE) and Spearman ρ correlation. We train three types of regressors: linear, quadratic, and tree-based via LightGBM [Ke et al., 2017]. Table 4.2 shows that the **quadratic model** fits the data the best. Hence, we use it as our PPM for subsequent experiments.

Simulation sample size selection. The PPM’s prediction time is significantly faster than conducting actual RM evaluations. As such, we can explore a large candidate dataset combination. To find out the optimal size of simulated candidates we evaluate the performance of PPM using $n \in \{128, 256, 512, 1024, 2048, 4096\}$ different candidates. This experiment suggests that performance plateaus around 1024 candidates (see Figure 4.3), achieving a score of 72.3%, indicating diminishing returns from larger candidate pools. Although the trend plateaus around 1024 candidates, we choose to balance between performance and run time and use 500 candidates in subsequent experiments.

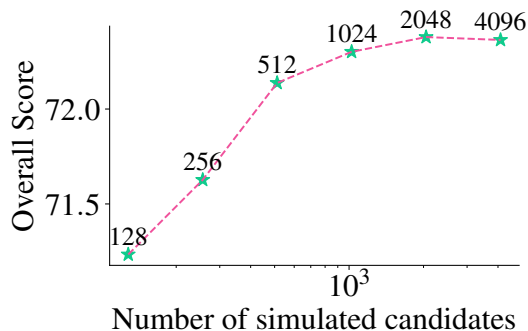


Figure 4.3: Actual RewardBench performance of the best configuration found given n simulated candidates.

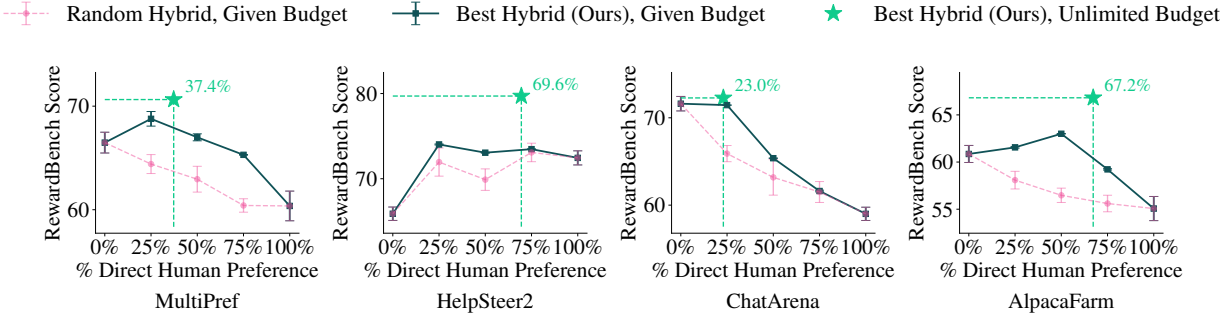


Figure 4.4: Comparison between HYPER and random selection given different annotation budgets on various preference datasets. The optimal budget and its corresponding performance is marked by a star (★). We report the average of the RewardBench score across three runs.

4.5.2 Generalization to Unseen Datasets

We next test whether the PPM trained on MULTIPREF generalizes to other unseen preference datasets. To do so, we apply the same routing strategy as described in §4.3.3. Instead of training separate PPMs for each unseen preference dataset, we only use a single PPM trained on MULTIPREF.

Datasets We use datasets with existing human preference annotations and augment them with LLM annotations from GPT-4 (gpt-4-turbo-2024-04-09) to simulate scenarios of routing a preference instance to a human annotator. These datasets include: Helpsteer2 [Wang et al., 2024c], AlpacaFarm Human Preferences [Dubois et al., 2023], and Chatbot Arena Conversations [Zheng et al., 2023a]. To control the effect of dataset size when comparing across datasets, we limit each preference mix to 7K instances after removing ties, the same size as MULTIPREF.

Baselines For each dataset, we use the following preference mixes to compare against our hybrid annotations: **100% Synthetic preference** containing purely synthetic preferences obtained from an LLM (see Appendix C.11 for more details on prompting GPT-4), **100% Direct Human Preference** with the human annotations of the dataset, and **25%, 50%, 75% Direct Human Preference** mixes (see Table C.5 in Appendix C.3) where we randomly swap a percentage of instances with human annotations while the rest are LM annotations. We train reward models based on the TULU 2 13B [Iverson et al., 2023] model on each of these mixes, and evaluate their performance on RewardBench.

RewardBench Performance										
Preference Mix	MULTIPREF (Appendix C.1)					Helpsteer2 [Wang et al., 2024c]				
	Overall	Chat	Chat-Hard	Safety	Reasoning	Overall	Chat	Chat-Hard	Safety	Reasoning
	% Direct Human for Best Hybrid: 37.4%					% Direct Human for Best Hybrid: 69.6%				
100% Human	60.4	89.1	37.8	71.6	42.9	72.4	90.6	60.7	68.0	76.7
100% Synth.	66.5	90.2	34.6	69.7	71.3	65.8	71.6	64.0	45.2	82.7
Best Hybrid	70.6	94.4	35.1	74.8	78.2	79.7	89.9	64.9	77.0	87.0
Preference Mix	AlpacaFarm [Dubois et al., 2023]					ChatArena [Zheng et al., 2023a]				
	Overall	Chat	Chat-Hard	Safety	Reasoning	Overall	Chat	Chat-Hard	Safety	Reasoning
	% Direct Human for Best Hybrid: 67.2%					% Direct Human for Best Hybrid: 23.0%				
100% Human	55.0	85.5	44.5	38.5	51.6	59.0	90.6	50.4	36.3	58.8
100% Synth.	60.9	87.2	41.4	56.1	58.5	71.6	93.5	50.2	69.4	73.2
Best Hybrid	66.8	94.5	50.8	58.1	63.8	72.2	94.7	51.3	67.6	75.1

Table 4.3: Comparison of full direct human preferences and synthetic preferences and the best hybrid preference mix given unlimited budget on RewardBench. Reporting the average of three runs.

Results Figure 4.4 shows the RewardBench score for each dataset on different human annotation budgets across four preference datasets. Results show that in the majority of annotation budgets, **hybrid annotations from HYPER outperform that of random sampling**. This suggests that combining annotations is expected to result in RMs that perform better than relying solely on annotations from a single source (human or LM), and the performance can improve with a better routing strategy. We also obtain the best hybrid mix with empirical optimal budget for any given preference dataset as shown in Table 4.3. We observe that **the best hybrid mix requires 20–70% of direct human annotations** in order to outperform a more costly 100% direct human annotation setup, depending on the dataset. Our best hybrid preference mix outperforms using 100% synthetic annotations, suggesting that collecting human annotations is still valuable as long as the preference instances routed to humans benefit from their annotations.

Furthermore, we observe that in general, **RMs trained on full synthetic preference annotations tend to perform better on RewardBench than 100% human annotations**, except for the Helpsteer2 dataset. We hypothesize this is due to the higher annotation quality by Helpsteer2’s data vendor (ScaleAI) and their aggressive data quality control where the authors filtered-out preference instances with low inter-annotator agreement and with noisy preference ratings. Nevertheless, our results in Figure 4.4 suggest that HYPER can still push this performance further by using just 70% human annotations. We also train a PPM using candidates generated from Helpsteer2, and observed similar trends when using routed annotations on other datasets (see Appendix C.10.2).

Preference Mix	RewardBench Performance on Helpsteer2									
	Llama 3.1 8B [Dubey et al., 2024]					Qwen 2.5 7B [Yang et al., 2024; Team, 2024]				
	Overall	Chat	Chat-Hard	Safety	Reasoning	Overall	Chat	Chat-Hard	Safety	Reasoning
100% Human	64.7	91.1	51.0	39.2	78.7	71.8	87.7	54.5	60.6	84.0
100% Synth.	60.6	90.5	33.8	48.8	69.4	69.7	89.1	54.8	56.9	82.2
Best Hybrid	72.4	94.7	47.6	71.4	76.2	72.4	87.4	55.6	63.1	83.6

Table 4.4: Comparison of full direct human preferences and synthetic preferences on the best hybrid preference mix given unlimited budget on RewardBench and different base models of Helpsteer2 [Wang et al., 2024c]. Reporting the average of three runs.

4.5.3 Generalization to Other Base Models

We also test whether the hybrid mix retains its competitive performance when trained on different models other than TULU 2 13B.

Setup To test model generalization, we train RMs using Llama 3.1 8B [Dubey et al., 2024] and Qwen 2.5 7B [Yang et al., 2024; Team, 2024] on hybrid mixes of the Helpsteer2 dataset and evaluate the resulting model on RewardBench. Similar to §4.5.2, these mixes were identified by a PPM trained on MULTIPREF’s features.

Results Table 4.4 shows that the hybrid annotations from HYPER outperforms 100% direct and 100% synthetic human preferences, consistent with our findings in Sec. 4.5.2. These results suggest that the preference annotations routed by HYPER are model-agnostic, as demonstrated by our experiments with models other than TULU 2 13B.

4.5.4 Generalization to other Evaluation Tasks

To test whether HYPER generalizes to new tasks other than RewardBench, we evaluate the models trained on hybrid datasets on other benchmarks.

Setup We follow the practice of Ivison et al. [2024] to convert several LM benchmarks into a “Best-of-N” reranking format for evaluating RMs: we sample 16 generations from the TULU-2 13B SFT model, score them using the testing reward models, and then use the top-scoring generation as the final output to compute the metrics. We evaluate on the following datasets: GSM8K [Cobbe et al., 2021b] for math, BIG-Bench Hard

Best-of-N Evaluation Performance												
Pref. Mix	MULTIPREF (Appendix C.1)						Helpsteer2 [Wang et al., 2024c]					
	Avg.	% Direct Human for Best Hybrid: 37.4%					Avg.	% Direct Human for Best Hybrid: 69.6%				
	GSM8K	BBH	IFEval	Codex	AlpacaEval		GSM8K	BBH	IFEval	Codex	AlpacaEval	
100% Human	48.3	38.0	47.3	43.1	24.4	88.6	52.6	52.3	51.0	45.8	26.2	87.7
100% Synth.	49.4	41.7	49.0	44.9	23.2	88.3	51.0	48.6	52.0	47.0	24.4	83.1
Best Hybrid	50.5	48.1	50.2	44.7	21.3	88.1	52.8	51.7	49.9	48.1	29.3	85.1
Pref. Mix	AlpacaFarm [Dubois et al., 2023]						ChatArena [Zheng et al., 2023a]					
	Avg.	% Direct Human for Best Hybrid: 67.2%					Avg.	% Direct Human for Best Hybrid: 23.0%				
	GSM8K	BBH	IFEval	Codex	AlpacaEval		GSM8K	BBH	IFEval	Codex	AlpacaEval	
100% Human	50.4	48.2	50.7	42.7	23.8	86.6	53.9	52.3	52.4	44.9	28.7	91.4
100% Synth.	53.1	52.3	52.6	44.7	26.2	89.6	53.7	54.0	52.3	44.5	26.8	90.9
Best Hybrid	53.3	53.5	52.7	45.5	23.8	91.0	52.8	51.9	51.8	44.5	25.0	90.8

Table 4.5: Comparison of full direct human preferences and synthetic preferences on the best hybrid preference mix given unlimited budget using Best-of-N evaluation.

(BBH) [Suzgun et al., 2022] for reasoning, IFEval [Zhou et al., 2023b] for precise instruction following, Codex HumanEval [Chen et al., 2021] for coding, and AlpacaEval [Li et al., 2023b] for general chat capabilities. Further information on the dataset setup can be found in Appendix C.6.

Results Table 4.5 shows the Best-of-N evaluation performance of the best hybrid mix found by our method. Our hybrid mix outperforms using either human or synthetic labels alone by 2–3% on three out of the four preference datasets. On Helpsteer2, 100% human labels perform better than 100% synthetic, while MULTIPREF and AlpacaFarm show the opposite trend, reflecting varying human annotation quality—our method demonstrates improvement in three cases despite this variation. ChatArena is the exception, where our method does not improve upon the original dataset. The trend in ChatArena’s Best-of-N performance differs from RewardBench, and we suspect that its due to its reliance on Internet volunteers with underspecified annotation guidelines. Further investigation of this discrepancy is left for future work.

4.6 Analysis: When are Human Annotations Helpful?

We investigate the features learned by the PPM to understand characteristics that render a preference instance a better fit for direct human annotation. To quantify the effect of routing an instance to human annotators, we compute its *expected performance gain* as described in §4.3.4. This analysis makes three key assumptions: (1) the performance gain from human annotation is linear; (2) samples are independent of each other, and (3) the PPM fits the data well. While the first two assumptions may not hold in general, they provide a tractable

Tag	Gain $\times 10^{-3}$	Tag	Gain $\times 10^{-3}$
BERTScore $\in [0.33, 0.67]$	0.19	Subject Of Expertise: Materials Science and Engineering	-0.00
Subject Of Expertise: Chemical Engineering	0.11	Subject Of Expertise: Library and Museum Studies	-0.10
Subject Of Expertise: Religion	0.09	Subject Of Expertise: Media Studies and Communicatio	-0.10
Safety Concern: Moderate	0.09	Subject of Expertise: Military Sciences	-0.10
Subject Of Expertise: Anthropology	0.06	Subject Of Expertise: Family And Consumer Science	-0.63

Table 4.6: Average gain in MULTIPREF’s performance when routing 100 random preference instances to a human annotator for each tag. Showing top- and bottom-five tags (See the full list in Appendix Table C.7).

framework for analyzing the relative importance of human annotation for different instances.

To estimate the performance gain of each tag $t \in T$, we route n instances that satisfy the tag’s condition (e.g., “BERTScore between two responses is $\in [0.33, 0.67]$ ”) and compute the gain Δ normalized on the count of instances with that tag. Table 4.6 shows the top- and bottom-five tags based on the performance gain. This list reveals that instances with moderate semantic similarity between responses (measured by BERTScore), moderate safety concern, and moderate complexity of intents tend to benefit more from direct human annotations. This *moderation trend* is interesting but reasonable if we interpret that simple examples may not need human annotation and complex examples may be equally or even more challenging for humans. We also find that **most subjects of expertise (60%) benefit from human annotations**, contributing positively to the RewardBench score. Preferences with prompts that require expert domain knowledge (Δ : 6.438E-6) to answer also benefit from human annotations as opposed to prompts requiring basic domain knowledge (Δ : -0.095E-6) or answerable by the general public (Δ : -0.050E-6).

4.7 Discussion & Future Work

Grounding of preference feedback quality. Quality control is critical for human data annotation, especially in the modern era of building LMs. Typically, researchers use agreement as a metric for quality. However, for preference annotation, early works all ended up with relatively low agreement between annotators or even between annotators and researchers [Bai et al., 2022a; Touvron et al., 2023a; Dubois et al., 2023]. This is largely due to the complexity of the tasks (e.g., many facts to verify, the expertise required, etc.), as well as the subjectivity in many cases (e.g., style preference, sensitive topics, safety threshold, etc.). This poses challenges for the data annotation process, as there is no ground truth for measuring the quality. In this work, we decide to ground the data quality into the model training performance (i.e., the utility of the data), and our

framework can optimize towards this goal. Future work can explore other downstream utility metrics for optimization.

Scaling the size of preference annotation. Although we show the successful generalization of our router when applying it to other preference datasets (§4.5.2), this set of experiments is done at the same size (7K after removing ties). It remains unclear how well our performance prediction model can extrapolate beyond the training data size and predict what instances can add performance gain after 7K, so that we can keep growing our preference data to a larger size. We believe our current results and the patterns we find (§4.6) can provide insights on how to save human efforts, but a systematic scaling of our framework may require future work.

Feedback beyond pairwise comparisons. We focus on pairwise preferences which compare overall model responses. However, several formulations of preference feedback exist such as fine-grained preferences [Wu et al., 2024], aspect-based preferences [Wang et al., 2023c, 2024c, also available in MULTIPREF] and preferences for process-reward models [Lightman et al., 2023; Uesato et al., 2022]. These annotations are more time consuming, hence, even more expensive, thus providing more room for leveraging LM annotation when possible. We leave this exploration for future work.

Generalization to downstream DPO / policy model performance. While hybrid preference annotations improve direct RM evaluation performance, it’s unclear if these gains extend to downstream tasks when training a DPO model or a policy model using PPO with the reward models. Ivison et al. [2024] found that improvements in reward models do not necessarily translate to improved downstream performance in PPO, as there are many confounding factors (e.g., the unlabeled prompts in PPO, the KL penalty, etc) that impact the PPO training. We tried testing the preference datasets using DPO (Appendix C.8) but only found small differences when switching datasets or the preference mixes. We hypothesize that downstream task performance is hard to measure (and still is an open research problem), and requires data collection at a larger scale to see significant effects.

Intra-group variability of annotators. One of our key assumptions is that there is no variability in intra-group annotators for both humans and LMs. When HYPER decides to route a preference instance to a human

or an LM, we don't make finegrained decisions as to what type of human annotator (or which LM) should annotate. However, we believe that MULTIPREF can enable this type of analyses especially for direct human feedback, as the dataset disambiguates between normal and expert crowdworker annotations. We leave this exploration for future work.

4.8 Summary

We introduce HYPHER, a framework that routes preference instances to either human annotators or to an LM that would maximize the performance of an RM trained on such hybrid-annotated data. Our results suggest that the hybrid mix from HYPHER outperforms all baseline annotation combinations on RewardBench, and that this trend generalizes to other models, benchmarks (via Best-of-N reranking), and unseen preference datasets. It also outperforms random sampling for a given set of human annotation budgets. Our analyses reveal that human annotations are most beneficial for instances with moderate response similarity and prompts in specific subject domains, among others. We hope HYPHER contributes to data-centric approaches in understanding human preferences and to more efficient preference collection methods in the future.

Chapter 5

Post-training LMs with Open Data

5.1 Overview

The training of large LMs like ChatGPT has converged into broadly two stages: pre-training on extensive text to grasp knowledge and skills, and post-training with various techniques to align with human intents and further improve. The post-training side, because of the complexity of training methods and many direct use cases, attracts significant attention, leading to many pieces of innovations, including datasets, algorithms, and evaluations. However, it is unclear what really excel, how to integrate them, and how they truly compare with proprietary models that are heavily engineered. Therefore, we have initiated and been working on the TULU series of work, which starts from open-weight pre-training models such as Llama and systematically studies the integration of post-training techniques.

This chapter details the first work in the TULU series, which focuses on the instruction tuning step. We provide a comprehensive evaluation of instruction-tuning resources: specifically, we conduct a large number of instruction-tuning experiments spanning a dozen public corpora, and models ranging in scale from 6.7B to 65 B. We evaluate both specific model capabilities (i.e., factual knowledge, reasoning, multilinguality, coding, safety) and open-ended instruction-following abilities. We report results based on automatic, model-based, and human-based evaluation metrics.

Our evaluation reveals that instruction tuning over different datasets appears to promote specific skills, and no one dataset provides the best performance across all evaluations. We also find that the underlying base

model is paramount, with better base models (whether it be models trained on more tokens or larger models) performing better across the board. Surprisingly, we also find that the best-performing models in model-based evaluation are not the same as those that perform best on benchmark-based automatic evaluations, potentially partially due to GPT-4’s strong bias toward long, diverse generations.

Building on our findings, we introduce TÛLU 🦙, a suite of 7B to 65B LLAMA models finetuned on a combination of data sources, leading to the best overall performance. TÛLU 🦙 65B is the largest publicly-released fully-instruction tuned LLAMA variant at the time of writing, to the best of the authors’ knowledge. It is trained on 7 popular available datasets, and yields the best average performance across most model sizes while remaining within 29% of the best-performing model on each individual task.

In summary, our key findings include:

- Instruction datasets targeted at specific domains and/or capabilities are extremely effective at improving model performance in those aspects.
- Larger or pretrained-for-longer base models consistently perform better than smaller ones after instruction tuning.
- Our model TÛLU 🦙 – fine-tuned LLAMA on a combination of existing instruction datasets – achieves the best average performance across benchmarks, although it is not the overall best when considering different evaluation settings independently.
- Even a very large (65B) model finetuned on a large mix of instruction datasets fails to outperform ChatGPT, although it does perform significantly better than similar smaller models.
- Model-based preference evaluation on open-ended instruction following correlates strongly with the average number of unique tokens generated by a model, suggesting that model-based preference evaluation has biases that may hide differences in model capabilities.

We open-source the code for training and evaluating these large language models. We also release checkpoints trained on the different instruction datasets and their mixtures, including TÛLU 🦙.¹ We hope this facilitates further development and investigation of open instruction-tuned models.

¹<https://github.com/allenai/open-instruct>

5.2 Background

Instruction Tuning. As explained in previous chapters, *instruction tuning*, in general, refers to the practice of finetuning pretrained language models to better understand and respond to a wide variety of human requests that are expressed in natural language [Mishra et al., 2022b; Wei et al., 2022a; Ouyang et al., 2022]. In particular, instruction tuning is concerned with requests that include some indication of the task to be performed within the request itself (e.g., including task instructions in the input prompt). It has arisen as a critical step for generalizing models to new scenarios without dedicated training, and for letting non-experts naturally interact with these models. The training paradigms of instruction tuning can vary from supervised learning using demonstrations [Wei et al., 2022a; Sanh et al., 2022; Wang et al., 2022b; Longpre et al., 2023a] to reinforcement learning from feedback data [Ouyang et al., 2022; Bai et al., 2022a]. In this work, we focus on the supervised learning setup, with a focus on other factors.

The success of instruction tuning requires at least two key components: 1) a powerful pretrained language model that has grasped a vast amount of knowledge from web-scale pretraining, and 2) an instruction dataset that is diverse and representative enough to adapt the LM to potential downstream usage. We study these two factors in this work and introduce our studied open resources below.

Instruction Datasets. We attempt to collect a representative sample of different styles of datasets (listed in Table 5.1), including datasets: (1) created by researchers from existing NLP datasets (SuperNI [Wang et al., 2022b], Flan V2 [Longpre et al., 2023a]); (2) written by humans from scratch for the purpose of instruction tuning (Dolly [Databricks, 2023], Open Assistant 1 [Köpf et al., 2023]); (3) generated by proprietary models (Self-Instruct [Wang et al., 2022a], Unnatural Instructions [Honovich et al., 2022a], Alpaca [Taori et al., 2023], Baize [Xu et al., 2023], GPT4-Alpaca [Peng et al., 2023]); (4) comprised of user-shared prompts accompanied by model-generated completions (ShareGPT² [Chiang et al., 2023]); (5) built for specific skills (CoT [Wei et al., 2022b] for chain-of-thought, Code-Alpaca [Chaudhary, 2023] for code generation). See Appendix D.1 for further details.

² ShareGPT (<https://sharegpt.com/>) data was used to build the Vicuna model [Chiang et al., 2023], but the exact dataset has not been released. We instead use a reproduced version from https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered/tree/main/HTML_cleaned_raw_dataset, and follow Vicuna to split the long conversations into blocks with a maximum length of 2048 tokens.

Datasets	Sourced from	# Instances	\bar{N}_{rounds}	\bar{L}_{prompt}	$\bar{L}_{completion}$
SuperNI [Wang et al., 2022b]	NLP datasets + Human-written Instructions	96,913	1.0	291.1	38.7
CoT [Wei et al., 2022b]	NLP datasets + Human-written CoTs	100,000	1.0	266.0	53.2
Flan V2 [Longpre et al., 2023a]	NLP datasets + Human-written Instructions	100,000	1.0	355.7	31.2
Dolly [Databricks, 2023]	Human-written from scratch	15,011	1.0	118.1	91.3
Open Assistant 1 [Köpf et al., 2023]	Human-written from scratch	34,795	1.6	34.8	212.5
Self-instruct [Wang et al., 2022a]	Generated w/ vanilla GPT3 LM	82,439	1.0	41.5	29.3
Unnatural Instructions [Honovich et al., 2022a]	Generated w/ InstructGPT ₀₀₂	68,478	1.0	107.8	23.6
Alpaca [Taori et al., 2023]	Generated w/ InstructGPT ₀₀₃	52,002	1.0	27.8	64.6
Code-Alpaca [Chaudhary, 2023]	Generated w/ InstructGPT ₀₀₃	20,022	1.0	35.6	67.8
GPT4-Alpaca [Peng et al., 2023]	Generated w/ InstructGPT ₀₀₃ + GPT4	52,002	1.0	28.0	161.8
Baize [Xu et al., 2023]	Generated w/ ChatGPT	210,311	3.1	17.6	52.8
ShareGPTsection 5.2	User prompts + outputs from various models	168,864	3.2	71.0	357.8

Table 5.1: Instruction datasets investigated in this work. CoT and FLAN V2 are sampled to 100K to match the sizes of other datasets. We report the average number of conversation turns (\bar{N}_{rounds}), average length of prompts (\bar{L}_{prompt}), average length of completion ($\bar{L}_{completion}$).

Pretrained Models. We primarily use the LLAMA suite [Touvron et al., 2023a,b], a series of pretrained models ranging in size from 6.7B to 65B parameters. We initially experimented with the LLAMA-1 models for the first version of this paper and added LLAMA-2 in our camera-ready, which uses similar numbers of parameters but were trained over significantly more tokens.

These models represent the largest, highest-quality pretrained models available to the community (albeit under restrictive licensing). We also consider OPT [Zhang et al., 2022] and Pythia [Biderman et al., 2023] models with a size comparable to the LLAMA 6.7B model, to examine the effect of different base models. For simplicity, we will round all the sizes to the nearest integer. We note several ongoing efforts to pre-train similar- or better-quality models [Geng and Liu, 2023; MosaicML, 2023; Almazrouei et al., 2023]. We believe our findings should hold for these models and future stronger open base models.

Base LMs	# Params	# Tokens
LLaMa [Touvron et al., 2023a]	6.7B	1.0T
	13.0B	1.0T
	32.5B	1.4T
	65.2B	1.4T
LLaMa-2 [Touvron et al., 2023b]	6.7B	2.0T
	13.0B	2.0T
OPT [Zhang et al., 2022]	6.7B	180B
Pythia [Biderman et al., 2023]	6.9B	300B

Table 5.2: Base models that we finetuned in this work.

5.3 Training Models with Various Datasets

5.3.1 Unifying the Format

We format all datasets to follow a chatbot-style schema to unify the varied styles and formats of the instruction datasets, shown in Figure 5.1. This allows us to fit arbitrary rounds of interactions between the user and the language model (a.k.a. “assistant”) into one input sequence and encode them together with a causal language model. We add special tokens `<|user|>` and `<|assistant|>` before user utterances and target assistant responses respectively, and an end-of-text marker `</s>` at the end of each assistant output, which, at inference time, will stop the model’s response for each round.

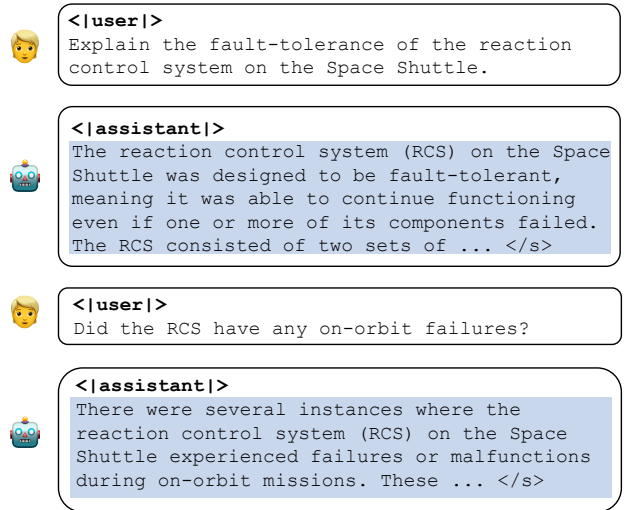


Figure 5.1: An example from ShareGPT data. We use `<|role|>` to set the boundary between messages. The entire sequence is encoded together, and loss is computed on the assistant parts (colored in blue).

5.3.2 Model Training Details

During training, we compute loss only on tokens after `<|assistant|>` and before the next `<|user|>` token. More formally, we consider an instruction dataset as consisting of N tuples, each with i turns, $\{(x_1^j, y_1^j, x_2^j, y_2^j, \dots, x_i^j, y_i^j)\}_{j=1}^N$, where x_i is a user prompt and y_i the desired output. For most instances, $i = 1$, and we train the model to output y^j given x^j . However, in the case of conversation datasets, we train the model to predict y_i^j given some conversation history $x_1^j, y_1^j, x_2^j, \dots, x_i^j$. We train decoder-only models, and use teacher-forcing with loss masking to train the models, where we mask all tokens belonging to the input sequence(s) x_i . Given X as the tokens belonging to the input, and Y as the target tokens, the loss function is:

$$L = - \sum_j \log p_\theta(t_j | t_{<j}) \times \begin{cases} 1 & \text{if } t_j \in Y \\ 0 & \text{otherwise} \end{cases}$$

where t_j is the j th input token (belonging to X or Y). See Appendix §D.2 for further training details.

5.3.3 TÛLU 🐪: a Better Instruction-Tuned Model by Combining Resources

Existing studies [Wang et al., 2022b; Longpre et al., 2023a] (and our own evaluation below) have shown that increasing the diversity of instructions can effectively improve the performance of instruction tuning. Following this motivation, we create two mixtures of datasets:

Human data mixture, which comprises the best human-authored datasets, including FLAN V2, CoT, Dolly, and Open Assistant 1 (we exclude SuperNI as FLAN V2 includes most tasks in SuperNI);

Human+GPT data mixture, which comprises the human mixture and three additional datasets that have generations by OpenAI GPT models, including GPT4-Alpaca, Code-Alpaca, and ShareGPT.

For both mixtures, we concatenate datasets and leave exploring more complex sampling mixtures to future work. We name LLAMA models trained on the Human+GPT data mixture TÛLU 🐪, after a hybrid camel resulting from interbreeding between different species. We differentiate the TÛLU models trained from the LLAMA-2 base models by versioning them as TÛLU-1.1.

5.4 Evaluation Setup

Evaluation of instruction-following models remains a challenging problem due to the enormous scope of “generality” and its open-ended nature. However, we argue that general-purpose models should be able to perform some core tasks before they can generalize to satisfy various practical needs. As such, we set up a multi-faceted evaluation to cover several key aspects of capabilities covering core abilities and open-ended instruction following. Our evaluations closely follow prior work on evaluating instruction-tuned models [Chung et al., 2022; Anil et al., 2023; Wang et al., 2022a; Chiang et al., 2023; Dubois et al., 2023], but serve as the first one to compile them together for systematic evaluation.

5.4.1 Facets of Evaluation

Factual knowledge is essential for language models to serve users’ information needs. We use the Massive Multitask Language Understanding dataset (MMLU [Hendrycks et al., 2020]) for measuring models’ factual knowledge. MMLU consists of a set of questions about 57 subjects ranging in difficulty from elementary levels to professional levels, and its multiple-choice format makes it suitable for probing models’ knowledge

without worrying about the open-endedness of generations.

Reasoning is another fundamental ability for models, especially for solving complex tasks. We use the test split of Grade School Math dataset (GSM Cobbe et al. [2021a]) to evaluate models’ mathematical reasoning capabilities. We also adopt Big-Bench-Hard (BBH [Suzgun et al., 2022]), which contains 23 challenging tasks from Big-Bench [Srivastava et al., 2022], to evaluate models’ general reasoning capabilities.

Multilinguality acts as an important perspective of models for serving people from different backgrounds. We use TyDiQA Clark et al. [2020], a multilingual question answering benchmark covering 11 typologically diverse languages for testing how much models can process non-English text. We use the gold-passage setup where one passage containing the reference answer is given.

Coding is a particular application that people have used language models for and might be important for integrating these models with external tools [Cai et al., 2023]. We use the HumanEval dataset Chen et al. [2021] to evaluate the models’ capability to generate functionally correct programs from docstrings. To avoid ambiguity with our human evaluation, we call this dataset Codex-Eval in this paper.

Open-ended instruction following. While the performance on the benchmarks above quantifies the models’ ability at specific skills, it may not reflect how well the models can handle instructions from real users, which cover highly diverse requests and are often open-ended. For example, the popular ShareGPT dataset contains instances of users asking for programming help, resume formatting tips, educational role-playing, pronunciation suggestion, fanfiction writing, and more. We evaluate such open-ended instructability of models using both model-based evaluation (§5.4.2) and human evaluation (§5.4.3), both of which consist of multiple test sets from existing studies Wang et al. [2022a]; Chiang et al. [2023]; Köpf et al. [2023]; Bai et al. [2022a]; Geng et al. [2023].

Safety is of particular concern regarding the fast-developing language models to ensure the ethical and proper use of them. Following LLAMA-2 [Touvron et al., 2023b], we employ ToxiGen [Hartvigsen et al., 2022] to measure the amount of toxic language and hate speech generation across different groups when the models are prompted to do so. We also adopt TruthfulQA [Lin et al., 2022] to measure how well models can avoid generating known falsehoods due to misconceptions or false beliefs while providing useful information.

For all the benchmark-based evaluations, we follow their standard metrics, while we subsample some benchmarks to a reasonable size to improve the efficiency of doing chain-of-thought reasoning. We refer the

reader to Appendix §D.3 for the setup details.

5.4.2 Model-Based Evaluation using GPT-4

To evaluate the open-ended instructability, we first adopt a model-based approach introduced in AlpacaEval [Li et al., 2023b]. The test set consists of 805 instructions, with 252 instructions from the Self-Instruct evaluation Wang et al. [2022a], 188 from the Open Assistant evaluation Köpf et al. [2023], 129 from the helpful evaluation by Anthropic Bai et al. [2022a], 80 from the Vicuna evaluation Chiang et al. [2023], and 156 from the Koala evaluation Geng et al. [2023].

We use their simulated GPT-4 annotator, which computes the win rate of the testing model as judged by GPT-4 when compared to the outputs produced by InstructGPT₀₀₃. We use the AlpacaEval codebase and prompts [Li et al., 2023b] to make our scores directly comparable to those on the AlpacaEval leaderboard³. When doing pairwise comparisons with GPT-4, the orders of model outputs are randomized to avoid position bias during evaluation [Wang et al., 2023a]. We do not evaluate vanilla LLAMA models due to them having little instruction-following ability without further prompt engineering.

5.4.3 Human Evaluation

To further test the quality of the open-ended generations, we conduct a human evaluation based on 332 instructions that combine the Self-Instruct evaluation set [Wang et al., 2022a] and Vicuna evaluation set [Chiang et al., 2023]. Inspired by Bai et al. [2022a], we design a similar interface (Figure B.2) for gathering human judgments of model outputs along the following dimensions. We note that we evaluated based on our fine-tuned LLAMA-1 models, as LLAMA-2 was not available at the time of this experiment.

Individual acceptability. We ask human raters to assess whether each system’s responses were acceptable in isolation. This is a binary decision, and we ask the raters to mark a response as acceptable if and only if the response answered the request in the query, had no significant errors, and did not have repetitive information.

Pairwise preference. We then ask humans to compare the outputs of two systems and select which one they think is more helpful. This is a 5-way decision, and the raters could select if one of the responses is “clearly” or “slightly” better than the other or if it is a tie that both responses were equally good or bad.

³https://tatsu-lab.github.io/alpaca_eval/

	MMLU (factuality)	GSM (reasoning)	BBH (reasoning)	TydiQA (multilinguality)	Codex-Eval (coding)	AlpacaEval (open-ended)	Average
	EM (0-shot)	EM (8-shot, CoT)	EM (3-shot, CoT)	F1 (1-shot, GP)	P@10 (0-shot)	Win % vs InstructGPT ₀₀₃	
Vanilla LLaMa 13B	42.3	14.5	39.3	43.2	28.6	-	-
+SuperNI	49.7	4.0	4.5	50.2	12.9	4.2	20.9
+CoT	44.2	40.0	41.9	47.8	23.7	6.0	33.9
+Flan V2	50.6	20.0	40.8	47.2	16.8	3.2	29.8
+Dolly	45.6	18.0	28.4	46.5	31.0	13.7	30.5
+Open Assistant 1	43.3	15.0	39.6	33.4	31.9	58.1	36.9
+Self-instruct	30.4	11.0	30.7	41.3	12.5	5.0	21.8
+Unnatural Instructions	46.4	8.0	33.7	40.9	23.9	8.4	26.9
+Alpaca	45.0	9.5	36.6	31.1	29.9	21.9	29.0
+Code-Alpaca	42.5	13.5	35.6	38.9	34.2	15.8	30.1
+GPT4-Alpaca	46.9	16.5	38.8	23.5	36.6	63.1	37.6
+Baize	43.7	10.0	38.7	33.6	28.7	21.9	29.4
+ShareGPT	49.3	27.0	40.4	30.5	34.1	70.5	42.0
+Human data mix.	50.2	38.5	39.6	47.0	25.0	35.0	39.2
+Human+GPT data mix.	49.3	40.5	43.3	45.6	35.9	56.5	45.2

Table 5.3: Comparison of different instruction tuning datasets, showing that different instruction-tuning datasets can excel in different aspects, and mixtures perform best on average.

	MMLU (factuality)	GSM (reasoning)	BBH (reasoning)	TydiQA (multilinguality)	Codex-Eval (coding)	AlpacaEval (open-ended)	Average
	EM (0-shot)	EM (8-shot, CoT)	EM (3-shot, CoT)	F1 (1-shot, GP)	P@10 (0-shot)	Win % vs InstructGPT ₀₀₃	
Pythia 6.9B	34.8	16.0	29.2	32.8	20.9	23.5	26.2
OPT 6.7B	32.6	13.5	27.9	24.1	8.9	25.9	22.2
LLAMA 7B	44.8	25.0	38.5	43.5	29.1	48.6	38.3
LLAMA-2 7B	49.2	37.0	44.2	52.8	33.9	57.3	45.7

Table 5.4: Performance of different base models after training on the Human+GPT data mixture.

To get a more reliable evaluation, we recruited a group of 18 expert annotators who are researchers at AI2 or students at UW. All of them are fluent English speakers, holding bachelor’s degrees or above.

5.5 Results

5.5.1 Analysis of Instruction Tuning Datasets and Base Models

To understand how the instruction datasets listed in Table 5.1 contribute to model abilities, we evaluated LLaMa 13B models trained on these datasets using our evaluation suite. Table 5.3 shows the results on our benchmark evaluation set, with more extensive results in App. D.4. We find that:

There is not a single best instruction tuning dataset across all tasks. Different datasets enable different capabilities in the model. Noteworthy examples include training on CoT being particularly helpful for mathematical reasoning in GSM and Code-Alpaca being helpful for Codex-Eval. We hypothesize that success on these tasks, which are significantly different from the rest of the evaluation tasks, calls for training sets where these tasks are well-represented. Apart from constructing task-specific datasets manually, distilling task-specific data from large models also appears to be an effective way to ensure this (e.g., CodeAlpaca is distilled from InstructGPT₀₀₃).

Combining datasets results in the best overall performance on the benchmark tasks. While models trained on our combination datasets are often not the best model for a single task (being the best only in 2 out of 6 evaluation settings), they are the best when measuring average performance across tasks. This suggests that future work into better dataset mixing or instruction-tuning modular models (e.g., mixture-of-experts [Shazeer et al., 2017]) is a promising direction for developing models that retain strong performance across all evaluation settings.

Base model quality is extremely important for downstream performance. We examine the impact of using different base models in Table 5.4, comparing LLAMA, OPT [Zhang et al., 2022], and Pythia [Biderman et al., 2023] models of comparable size trained on the Human+GPT data mix. Across all evaluation settings, we find that using LLAMA performs best by a significant margin, likely due to the fact that LLAMA is pretrained on significantly more tokens than the other models (see Table 5.2). This suggests that models pretrained on larger (or potentially higher-quality) corpora are preferable as base models for instruction tuning. The later addition of LLAMA-2 confirms this finding by showing a significant improvement can come from only the base model upgrade.

Some datasets degrade vanilla model performance. Notably, most datasets we evaluate cause degradation in performance on GSM and TydiQA over the vanilla base model. We hypothesise this is due to data style and quality. Many of the datasets we examine contain little to no examples of chain-of-thought-style reasoning and contain little to no multilingual data. As such, training on these datasets likely results in some forgetting of the CoT or multilingual abilities previously held by the model, resulting in degraded performance. Additionally, we note that self-instruct appears to cause degradations across most tasks, which we hypothesise is due to the relatively poor quality of the original self-instruct data, being generated by a









	MMLU (factuality)	GSM (reasoning)	BBH (reasoning)	TydiQA (multilinguality)	Codex-Eval (coding)	AlpacaEval (open-ended)	Average
	EM (0-shot)	EM (8-shot, CoT)	EM (3-shot, CoT)	F1 (1-shot, GP)	P@10 (0-shot)	Win % vs InstructGPT ₀₀₃	
Vanilla LLaMa models ↓							
LLaMa 7B	31.5	10.0	33.0	38.4	20.5	-	-
LLaMa 13B	42.3	14.5	39.3	43.2	28.6	-	-
LLaMa 30B	54.6	36.0	49.5	55.3	42.8	-	-
LLaMa 65B	58.7	50.0	58.1	56.8	46.9	-	-
LLaMa-2 7B	41.8	12.0	39.3	51.2	26.8	-	-
LLaMa-2 13B	52.0	25.0	48.9	56.5	32.5	-	-
65B models trained on alternate data mixtures ↓							
ShareGPT 65B	61.3 (+2.6)	59.0 (+9.0)	55.8 (-2.3)	31.6 (-25.2)	56.2 (+9.3)	73.6	56.3
Human mix. 65B	60.4 (+1.7)	60.0 (+10.0)	54.8 (-3.3)	58.3 (+1.7)	44.6 (-2.3)	43.4	53.6
 models trained on our final Human+GPT data mixture ↓							
TÜLU  7B	44.8 (+13.3)	25.0 (+15.0)	38.5 (+5.5)	43.5 (+5.1)	29.1 (+8.6)	48.6	38.3
TÜLU  13B	49.3 (+7.0)	40.5 (+26.0)	43.3 (+4.0)	45.6 (+2.4)	35.9 (+7.3)	56.5	45.2
TÜLU  30B	57.7 (+3.1)	53.0 (+17.0)	51.9 (+2.4)	51.9 (-3.4)	48.0 (+5.2)	62.3	54.1
TÜLU  65B	59.2 (+0.5)	59.0 (+9.0)	54.4 (-3.7)	56.6 (-0.2)	49.4 (+2.5)	61.8	56.7
 models trained on our final Human+GPT data mixture using LLAMA-2 ↓							
TÜLU-1.1  7B	49.2 (+7.4)	37.0 (+25.0)	44.2 (+4.9)	52.8 (+1.6)	33.9 (+7.1)	57.3	45.7
TÜLU-1.1  13B	52.3 (+0.3)	53.0 (+28.0)	50.6 (+1.7)	58.8 (+2.3)	38.9 (+7.4)	64.0	52.9
Proprietary models ↓							
ChatGPT	67.9	76.0	66.1	51.9	88.4	83.6	72.3
GPT-4	82.4	92.5	88.0	70.8	94.1	93.5	86.9

Table 5.5: Performance of TÜLU and other of our trained models to vanilla LLAMA models and the state-of-the-art proprietary models across evaluation settings. See Table D.1 for a complete list.

weaker model (base GPT-3) than the other GPT-distilled datasets.

5.5.2 Pushing the Limits of Open Models

Having established that (a) using a broad mix of data is best, and (b) using LLAMA as the base model is preferable to other open alternatives, we compare the performance of models trained on the Human+GPT data mix (TÜLU models) across all LLAMA sizes in Table 5.5. We find that:

Instruction tuning brings large benefits on top of LLAMA models at all sizes. On average, all LLAMA models improve considerably after instruction tuning.

Smaller models benefit most from instruction tuning. We find that relative improvements from instruction tuning are largest for the smallest models, and shrink as models get larger. Notably, the 65B

LLAMA model performs comparably or better than the 65B TüLU model on MMLU, BBH, and TydiQA. This suggests that **instruction-tuning does not help to enhance strong capabilities already present in the original model**, and also highlights that care must be taken during finetuning to avoid forgetting the base model’s original capabilities.

TüLU still lags behind state-of-the-art proprietary models. Despite the impressive performance of TüLU 65B, it lags behind ChatGPT and GPT-4 in all evaluation settings, contrary to prior claims that models trained on these open resources can match ChatGPT [Zhou et al., 2023a; Chiang et al., 2023]. We note **we cannot discount the possibility that either ChatGPT or GPT-4 was trained on significant portions of our evaluation suite**. However, the presence of a significant gap between TüLU models and ChatGPT matches our findings in the model and human-based evaluations, which are less likely to be compromised.

5.5.3 Evaluation of Potential Risks and Harms

We evaluate our models on ToxiGen and TruthfulQA to measure the degree to which different datasets are likely to yield models that generate toxic language or misinformation. We find that:

Trends remain similar to capability-focused benchmarks. Similarly to the results in Sec. 5.4.1, we find that GPT-distilled datasets yield the best overall performance and that there is a large variance in performance across datasets.

Models trained on GPT-sourced data yield less toxic generations than GPT. Larger models trained on GPT-distilled data appear to refuse to produce toxic generations almost entirely, despite the fact that ChatGPT and GPT-4 produce toxic generations a non-trivial amount of the time. We hypothesise this is due to our models overfitting on refusal-style behaviour, refusing to generate anything moderately toxic, while GPT models balance refusal behaviour with helpfulness to a greater extent.

Model ↓	ToxiGen (↓)		TruthfulQA (↑)	
	7B	13B	7B	13B
LLAMA	85.4	82.6	26.2	23.6
+ SuperNI	85.3	77.3	26.7	26.2
+ CoT	63.0	43.9	35.1	35.5
+ Flan V2	77.5	61.4	33.2	33.4
+ Dolly	72.1	78.9	30.1	32.9
+ Open Assistant 1	39.2	5.2	40.9	48.6
+ Self-instruct	89.0	89.3	22.4	22.4
+ Unnatural Inst.	35.8	55.7	27.3	31.7
+ Alpaca	63.2	58.1	33.5	39.8
+ Code-Alpaca	84.3	92.0	25.1	26.7
+ GPT4-Alpaca	3.9	1.2	51.2	56.7
+ Baize	77.2	41.2	42.4	43.9
+ ShareGPT	5.5	2.5	45.3	60.0
+ Human mix.	51.8	76.9	34.1	32.1
+ TüLU 🦙	10.6	0.1	44.6	41.6
ChatGPT	27.7		75.2	
GPT-4	10.6		82.3	

Table 5.6: Performance of models on ToxiGen (% toxic generations, lower is better) and TruthfulQA (% truthful and informative answers, higher is better). See Table D.2 and D.3 for the full breakdown.

Training Dataset ↓	7B	13B	30B	65B
SuperNI	2.9	4.2		
CoT	5.0	6.0		
Flan V2	3.1	3.2		
Dolly	11.0	13.7		
Open Assistant 1	51.4	58.1		
Self-instruct	4.0	5.0		
Unnatural Instructions	7.5	8.4		
Alpaca	21.4	21.9		
Code-Alpaca	15.3	15.8		
GPT4-Alpaca	57.3	63.1		
Baize	20.0	21.9		
ShareGPT	62.4	70.5	69.1	73.6
Human mix.	28.7	35.0	38.3	43.4
TÜLU 🦉	48.6	56.5	62.3	61.8

Table 5.7: Win-rate (%) of LLAMA models of varying sizes finetuned on the given dataset against InstructGPT₀₀₃ using AlpacaEval [Li et al., 2023b].

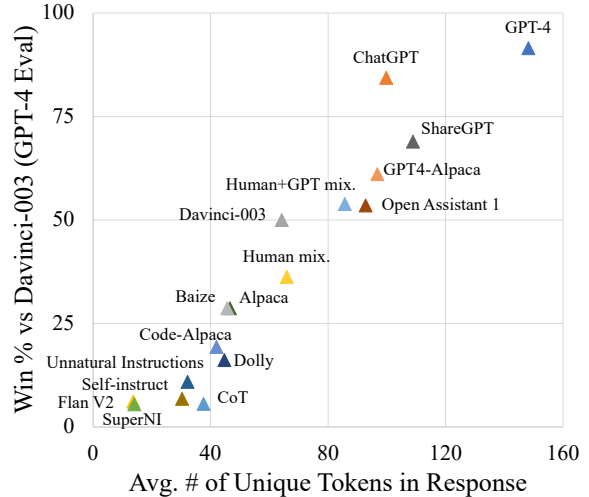


Figure 5.2: Win-rate scores of 13B models (trained on different datasets) given by GPT-4 strongly correlate with the average numbers of unique tokens in the model responses (Pearson $r = 0.96$).

TruthfulQA performance does not scale. Unlike other

benchmarks, we find that TruthfulQA performance does not improve with model size. Further examining this, we find that larger models do output more correct facts, but also tend to hedge and refuse to give informative answers more often, resulting in little to no overall improvements as model size increases.

5.5.4 Model-Based Evaluation Results for Open-Ended Generation

We report the AlpacaEval win-rates of our models in Table 5.7. We find that:

Models trained on mixtures based on traditional NLP datasets perform poorly. CoT, FLAN, and SuperNI all perform extremely poorly in open-ended instruction following, despite these datasets providing large improvements to the model capabilities tested in Table 5.3.

Datasets that encourage long, diverse generations perform best. Intrigued by ShareGPT’s performance, we plot the average number of unique tokens in model generations against the AlpacaEval win-rate in Figure 5.2. We find that the evaluation is **strongly correlated with the average number of unique tokens** (Pearson correlation of 0.96, $p \ll 0.05$). Given GPT-4’s strong performance on other tasks, we do not believe that GPT-4 evaluation is merely counting unique tokens, but this result highlights how model preference scores do not necessarily reward only model capabilities.

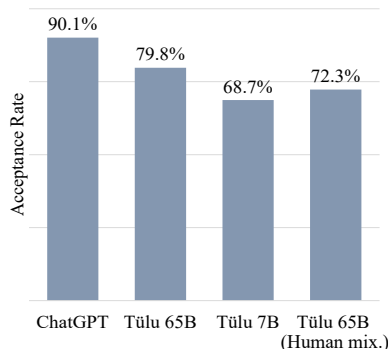


Figure 5.3: Human acceptance rates for four evaluated models.

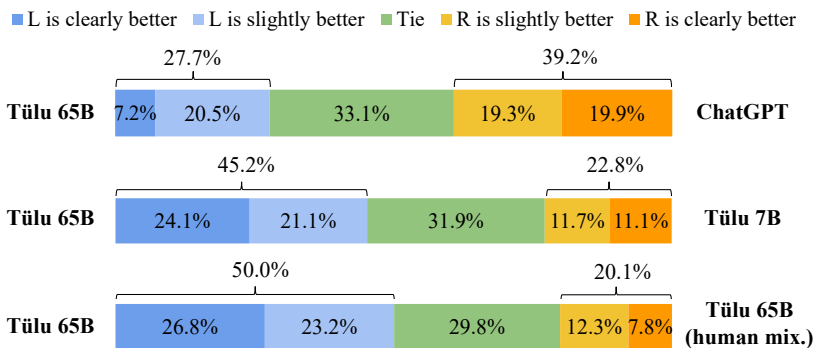


Figure 5.4: Human preference rates for three comparison pairs of models.

ShareGPT performs best. We find that ShareGPT consistently performs best across all model sizes, including models trained on data mixes that include ShareGPT. Models trained on ShareGPT achieve higher win-rates than models over twice their size (e.g., 13B ShareGPT vs 65B TüLU). We hypothesize this is due to ShareGPT’s diversity, size, and the high average # tokens of target responses.

Overall, these results suggest that while model preference evaluation is important, it does not provide a *holistic* evaluation of these models. Instead, model preference evaluation should only be included as part of a larger, more comprehensive evaluation setup.

5.5.5 Human Evaluation Results for Open-Ended Generation

Finally, we show the human evaluation results in Figure 5.4 and we refer the reader to Appendix §D.5.2 for the inner-annotator agreement. We find that the **human evaluation results largely correlate with the AlpacaEval and benchmark-based evaluation**: all evaluations show that 65B TüLU outperforms 7B TüLU, suggesting making use of larger base models is important, and there is still a nontrivial gap in performance between 65B TüLU and ChatGPT. We also find that **making use of distilled datasets provides a large performance boost**, suggesting that human-authored datasets are lacking in comparison. These observations are also consistent with the acceptability scores in Figure 5.3. However, we note that 7B TüLU outperforms the human-mix 65B TüLU in the model preference evaluation, but if we compare the acceptability scores in Figure 5.3, the opposite appears true. This is further evidence that model pairwise evaluation may not always reveal model deficiencies. In this case, the 65B human-mix model is more likely to yield acceptable (if not high-quality) responses than the 7B model.

5.6 Discussion & Future Work

Despite the comprehensiveness of our evaluations, we note that we did not exhaustively cover all possible evaluations: for example, we do not explicitly evaluate models on their multi-turn dialogue abilities nor their summarization abilities. Instead, we focus on a core set of capabilities we believe important, and cover broad open-ended tasks via our model and human preference-based evaluations.

We also note that we do not cover all possible instruction datasets and open models released recently, due to the computational cost of doing this. Instead, we focus on a wide set of datasets we believe are broadly representative of the type of open instruction datasets available (human-authored, skill-targeted, GPT-distilled, etc), and focused on the strongest base model widely available when performing experiments. Future work could investigate whether more recent strong base models (e.g., the Falcon model [Almazrouei et al., 2023]), or other instruction datasets, perform significantly better or differently from the models explored in this work.

Finally, we note that open-ended instruction-based evaluation is highly subjective and difficult due to its extremely open-ended nature. There is likely no one answer that is definitively the best for any given query, and different annotators (whether they be human or model) will have different biases and preferences. We also note that in the case of model-based evaluations, we primarily compare our model outputs to InstructGPT₀₀₃ generations, which may result in overly rewarding models that avoid shortcomings of InstructGPT₀₀₃, or not properly rewarding models that share strengths with InstructGPT₀₀₃.

5.7 Summary

In this work, we provide an extensive evaluation of a wide variety of publicly available resources for instruction-tuning models, and compare them to the strongest proprietary models currently available. We find that using strong base models is vital to performance, combining datasets works best on average (but does result in slight performance drops compared to best performance in specific tasks), and our strongest open models do not yet match ChatGPT or GPT-4. Furthermore, we believe that our evaluation highlights the need for the continued development of strong base models and broader, diverse datasets. Finally, we hope that our evaluation and released code and models enable more comprehensive evaluations and spur research to close these gaps and shed insights on all large language models, closed or open.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This dissertation addressed the critical challenge of steering general-purpose language models (LMs) to effectively serve a multitude of complex downstream applications. I have argued that moving beyond simply scaling pretraining data requires a systematic rethinking of how data is structured, created, shared, and utilized in the post-training phase. The core contribution of this work is the proposal and investigation of four scalable data paradigms designed to provide a more principled, efficient, and sustainable foundation for developing capable and aligned LMs.

First, the **unified data paradigm**, explored through instruction tuning and the development of the SUPER-NATURALINSTRUCTIONS benchmark and T_k -INSTRUCT model, demonstrated that representing diverse NLP tasks via natural language instructions significantly enhances model generalization. Second, the **synthetic data paradigm**, exemplified by the SELF-INSTRUCT framework, showed that LMs themselves can be leveraged to generate vast quantities of diverse instructional data, enabling scalable data creation and even model self-improvement. Third, the **human-AI collaboration paradigm**, through the HYPER framework, offered a novel approach to optimize preference data collection by intelligently routing annotation tasks, thereby improving the efficiency and quality of data for reinforcement learning from human feedback. Finally, the **open data paradigm**, investigated through the systematic study and development of the TULU models, highlighted the crucial role of open datasets and collaborative efforts in advancing research and democratizing

access to powerful LMs.

Collectively, these paradigms—unifying task representations, synthesizing training data with models, optimizing human-AI annotation, and fostering open data ecosystems—represent significant strides towards more robust and scalable methods for post-training LMs. The findings and methodologies presented herein have contributed to the evolving landscape of AI, shaping how we approach data in the ongoing quest for more capable, reliable, and adaptable language models. As AI continues to advance, a data-centric perspective remains paramount for sustainable progress.

6.2 Future Work

The research presented in this dissertation opens up several promising avenues for future investigation, building upon the data foundation established. I envision a future where AI systems advance in capability and generality while operating within an ecosystem that is ethically sound, legally clear, open, and beneficial to all. My future research agenda plans to enhance and expand this data foundation through four interconnected directions.

6.2.1 Data Optimization and Expansion at Scale

As the link between data and model performance becomes increasingly evident, research on principled data expansion and automated optimization is urgently needed. Current practices for expanding datasets are often driven by intuition rather than principled analysis. Future work must establish systematic approaches to the critical question: "Where should we expand new data?" This involves identifying gaps in model capabilities and strategically sourcing data to fill them. Such a strategy must move beyond simple data selection or pruning and toward a holistic optimization framework. This framework will need to decide what data to use, how to augment it, and where to source it from—be it human annotators or model synthesis—while considering the potential for errors and biases from each source. The algorithm presented in §4 on hybrid preference learning [Miranda et al., 2024] and my previous work on data cartography [Swayamdipta et al., 2020] can serve as initial attempts for more general methods.

6.2.2 Automated Verification for Model Improvement

As the downstream tasks and applications for AI systems become increasingly complex, it will be very costly and challenging for humans to supervise model responses entirely or to provide direct judgment on their correctness and quality. Developing automated verification systems is a crucial area of research. The goal is to build a versatile system capable of verifying diverse aspects of model outputs, such as factual correctness, logical consistency of reasoning, adherence to complex instructions, and safety guidelines. Such a system would integrate various techniques, including retrieval-augmented fact checking, tool use, reasoning, and formal methods. Building such systems will be valuable in multiple aspects. The verification systems can operate independently to provide more advanced evaluations or enhance user trust in the model outputs by presenting the verification results. More importantly, these verification systems must be integrated back into the training loop. Verified correct behaviors can act as positive examples, while identified failures provide negative signals or penalties in a "Reinforcement Learning from Verifiable Rewards" (RLVR) framework introduced in our TüLU 3 paper [Lambert et al., 2024a].

6.2.3 Fair and Responsible Data Practices in AI

Addressing the pressing issues around privacy, copyright, and ethical integrity in data usage is paramount for the sustainable development of AI. I plan to pursue two primary lines of research toward this goal. The first is developing robust and scalable methods for data attribution, tracing model outputs back to training data to ensure fair compensation and credit for original creators. This involves advancing techniques like influence functions [Koh and Liang, 2017] and watermarking [Kirchenbauer et al., 2023] to make them reliable investigative tools. OLMoTrace [Liu et al., 2025] has made initial efforts to efficiently track model responses to pretraining and finetuning data using efficient n-gram matching algorithms. The second is establishing better mechanisms for tracking the provenance of data to increase legitimacy, prevent data contamination, and ensure license compliance [Longpre et al., 2023b]. While we made valuable attempts in TüLU3 by manually checking the provenance of each data source [Lambert et al., 2024a], they are not scalable. Tackling these challenges will require interdisciplinary collaboration between computer science, law, and policy, building upon the transparent artifacts released from open projects like OLMo [Groeneveld et al., 2024] and TüLU [Wang et al., 2023b; Ivison et al., 2023, 2024; Lambert et al., 2024a].

6.2.4 Data for Real-World Multimodal Interactions

As the field is working on expanding the generality of AI systems, I believe that incorporating more comprehensive inputs into our models and teach the model how to act in the real world would be the critical next step. This necessitates a focus on data involving multiple modalities and real-world interactions, which is often sparse and raises significant concerns regarding privacy and safety. My research will advance this frontier by extending the unified view of tasks from my earlier work to formulate a new paradigm of tasks incorporating multi-modal inputs and real-world interactions based on existing work [Talmor et al., 2021; Hu et al., 2023]. Furthermore, my experience with synthetic textual data [Wang et al., 2022a] can inform the use of simulation to produce diverse interaction data at scale. Experiences from building open language models [Groeneveld et al., 2024] can also guide the open development of models that encompass modalities beyond language and vision, fostering collaboration with experts in perception, robotics, and planning.

Bibliography

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*.

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. 2024. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. [Falcon-40B: an open large language model with state-of-the-art performance](#). Huggingface Model Release.

Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Emilie Devijver, and Yury Maximov. 2022. [Self-training: A survey](#). *arXiv preprint arXiv:2202.12040*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q Tran, Dara Bahri, Jianmo Ni, et al. 2022. [ExT5: Towards Extreme Multi-Task Scaling for Transfer Learning](#). In *International Conference on Learning Representations (ICLR)*.

Stephen H Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht

- Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, et al. 2022. [PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts](#). In *Annual Meeting of the Association for Computational Linguistics (ACL) - System Demonstrations*.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022b. Constitutional AI: Harmlessness from AI Feedback.
- Michele Banko and Eric Brill. 2001. [Scaling to Very Very Large Corpora for Natural Language Disambiguation](#). In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. [Beat the ai: Investigating adversarial human annotation for reading comprehension](#). *Transactions of the Association for Computational Linguistics (TACL)*, 8:662–678.
- Mohaddeseh Bastan, Mahnaz Koupaee, Youngseo Son, Richard Sicoli, and Niranjan Balasubramanian. 2020. [Author’s sentiment prediction](#). In *International Conference on Computational Linguistics (COLING)*.
- Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2008. [The sixth pascal recognizing textual entailment challenge](#). In *Text Analysis Conference (TAC)*.
- Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A

suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Alexandra Birch, Miles Osborne, and Philipp Koehn. 2008. Predicting success in machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 745–754.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. [On the opportunities and risks of foundation models](#). *arXiv:2108.07258 [cs]*. ArXiv: 2108.07258.

Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. [Flex: Unifying evaluation for few-shot nlp](#). In *Advances in Neural Information Processing Systems* (NeurIPS).

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind

- Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, and et al. 2020a. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems* (NeurIPS).
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, and et al. 2020b. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems* (NeurIPS).
- Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2023. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*.
- Rich Caruana. 1997. [Multitask learning](#). *Machine learning*, 28(1):41–75.
- Sahil Chaudhary. 2023. [Code alpaca: An instruction-following llama model for code generation](#). GitHub repository.
- Kushal Chawla, Jaysa Ramirez, Rene Clever, Gale Lucas, Jonathan May, and Jonathan Gratch. 2021. [CaSiNo: A Corpus of Campsite Negotiation Dialogues for Automatic Negotiation Systems](#). In *Conference of the North American Chapter of the Association for Computational Linguistics* (NAACL).
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality](#). Blog post.
- Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. [Chatbot arena: An open platform for evaluating llms by human preference](#).
- Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30.

- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.
- Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. [TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages](#). *TACL*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021a. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021b. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- David Cohn, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. *Machine learning*, 15:201–221.
- Ronan Collobert and Jason Weston. 2008. [A unified architecture for natural language processing: Deep neural networks with multitask learning](#). In *International Conference on Machine Learning (ICML)*.
- Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. [The pascal recognising textual entailment challenge](#). In *Machine Learning Challenges Workshop*.
- Databricks. 2023. [Free dolly: Introducing the world’s first truly open instruction-tuned llm](#). Blog post.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*.

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.
- Jingfei Du, Édouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. 2021. [Self-training improves pre-training for natural language understanding](#). In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies*, pages 5408–5418.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. AlpacaFarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*.
- Avia Efrat and Omer Levy. 2020. [The Turking Test: Can Language Models Understand Instructions?](#) *arXiv preprint arXiv:2010.11982*.
- Logan Engstrom, Axel Feldmann, and Aleksander Madry. 2024. DsDm: Model-aware dataset selection with datamodels. *arXiv preprint arXiv:2401.12926*.
- Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. [A survey of data augmentation approaches for nlp](#). In *Annual Meeting of the Association for Computational Linguistics (ACL) ACL-IJCNLP - Findings*, pages 968–988.
- Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. [Speaker-follower models for vision-and-language navigation](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Nancy Fulda, Nathan Tibbetts, Zachary Brown, and David Wingate. 2017. [Harvesting common-sense navigational knowledge for robotics from uncurated text corpora](#). In *Conference on Robot Learning (IJCAI)*.

- Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. [Koala: A dialogue model for academic research](#). Blog post.
- Xinyang Geng and Hao Liu. 2023. [Openllama: An open reproduction of llama](#). GitHub Repository.
- Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah Smith, and Hannaneh Hajishirzi. 2024. [OLMo: Accelerating the science of language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand. Association for Computational Linguistics.
- Aditya Gupta, Jiacheng Xu, Shyam Upadhyay, Diyi Yang, and Manaal Faruqui. 2021. [Disfl-qa: A benchmark dataset for understanding disfluencies in question answering](#). In *Annual Meeting of the Association for Computational Linguistics (ACL) - Findings*.
- Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. [TOXIGEN: Controlling Language Models to Generate Implied and Adversarial Toxicity](#). In *ACL*.
- Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. [A joint many-task model: Growing a neural network for multiple nlp tasks](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Junxian He, Jiatao Gu, Jiajun Shen, and Marc’Aurelio Ranzato. 2019. [Revisiting self-training for neural sequence generation](#). In *International Conference on Learning Representations (ICLR)*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*.

William Hersh, Chris Buckley, TJ Leone, and David Hickam. 1994. [Ohsumed: An interactive retrieval evaluation and new large test collection for research](#). In *Conference of the Association for Computing Machinery Special Interest Group in Information Retrieval (SIGIR)*.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. 2017. [Deep Learning Scaling is Predictable, Empirically](#). *arXiv preprint arXiv:1712.00409*.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. [Distilling the knowledge in a neural network](#). In *Advances in Neural Information Processing Systems (NeurIPS) Workshop on Deep Learning*.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022a. [Unnatural instructions: Tuning language models with \(almost\) no human labor](#). *arXiv preprint arXiv:2212.09689*.

Or Honovich, Uri Shaham, Samuel R Bowman, and Omer Levy. 2022b. [Instruction induction: From few examples to natural language task descriptions](#). *arXiv preprint arXiv:2205.10782*.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. 2023. [Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering](#). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20406–20417.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. [Large language models can self-improve](#). *arXiv preprint arXiv:2210.11610*.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. 2022. [Datamodels: Predicting predictions from training data](#). *arXiv preprint arXiv:2202.00622*.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A Smith, Yejin Choi, and Hannaneh Hajishirzi. 2024. [Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback](#). *arXiv preprint arXiv:2406.09279*.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. 2023. [Camels in a changing climate: Enhancing LM adaptation with Tulu 2](#). *arXiv preprint arXiv:2311.10702*.

- Nan-Jiang Jiang and Marie-Catherine de Marneffe. 2022. [Investigating reasons for disagreement in natural language inference](#). *Transactions of the Association for Computational Linguistics*, 10:1357–1374.
- Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2022. [Large language models struggle to learn long-tail knowledge](#). *arXiv preprint arXiv:2211.08411*.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Neural Information Processing Systems*.
- Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. [UnifiedQA: Crossing Format Boundaries With a Single QA System](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP) - Findings*.
- Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. [Natural language to sql: Where are we today?](#) *Proceedings of the VLDB Endowment*, 13(10):1737–1750.
- John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.
- Hannah Kirk, Andrew Bean, Bertie Vidgen, Paul Rottger, and Scott Hale. 2023. [The past, present and better future of feedback learning in large language models for subjective human preferences and values](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2409–2430, Singapore. Association for Computational Linguistics.
- Hannah Rose Kirk, Alexander Whitefield, Paul Röttger, Andrew Bean, Katerina Margatina, Juan Ciro, Rafael Mosquera, Adina Williams, He He, Bertie Vidgen, and Scott Hale. 2024. The PRISM Alignment Project: What Participatory, Representative and Individualised Human Feedback Reveals About the Subjective and Multicultural Alignment of Large Language Models. *arXiv preprint arXiv:2404.16019*.
- Nikita Kitaev, Steven Cao, and Dan Klein. 2019. [Multilingual constituency parsing with self-attention and pre-training](#). In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3499–3505.

- Nikita Kitaev and Dan Klein. 2018. [Constituency parsing with a self-attentive encoder](#). In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2676–2686.
- Pang Wei Koh and Percy Liang. 2017. [Understanding black-box predictions via influence functions](#). *arXiv:1703.04730 [cs, stat]*. ArXiv: 1703.04730.
- Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. 2023. Openassistant conversations—democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024a. TULU 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.
- Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. 2024b. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*.
- Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang, and Felix Hill. 2022. [Can Language Models Learn from Explanations in Context?](#) *arXiv preprint arXiv:2204.02329*.
- Logan Lebanoff, John Muchovej, Franck Dernoncourt, Doo Soon Kim, Lidan Wang, Walter Chang, and Fei Liu. 2020. [Understanding points of correspondence between sentences for abstractive summarization](#). In *Annual Meeting of the Association for Computational Linguistics (ACL) - Student Research Workshop*.
- Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2023. RLAIIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. *arXiv preprint arXiv: 2309.00267*.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

- Minzhi Li, Taiwei Shi, Caleb Ziems, Min-Yen Kan, Nancy Chen, Zhengyuan Liu, and Diyi Yang. 2023a. [CoAnnotating: Uncertainty-guided work allocation between human and large language models for data annotation](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1487–1505, Singapore. Association for Computational Linguistics.
- Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.
- Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023c. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*.
- Chin-Yew Lin. 2004. [ROUGE: A package for automatic evaluation of summaries](#). In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252.
- Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2022. [WANLI: Worker and ai collaboration for natural language inference dataset creation](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP) - Findings*.
- Jiacheng Liu, Taylor Blanton, Yanai Elazar, Sewon Min, YenSung Chen, Arnavi Chheda-Kothary, Huy Tran, Byron Bischoff, Eric Marsh, Michael Schmitz, et al. 2025. Olmotrace: Tracing language model outputs back to trillions of training tokens. *arXiv preprint arXiv:2504.07096*.
- Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and

- Min Lin. 2024. Regmix: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*.
- Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023a. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.
- Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetla, et al. 2023b. The data provenance initiative: A large scale audit of dataset licensing & attribution in ai.
- Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, and Chang Zhou. 2023. # instag: Instruction tagging for diversity and complexity analysis. *arXiv preprint arXiv:2308.07074*.
- Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. [Teaching small language models to reason](#). *arXiv preprint arXiv:2212.08410*.
- Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. [The natural language decathlon: Multitask learning as question answering](#). *arXiv preprint arXiv:1806.08730*.
- Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282.
- Dheeraj Mekala, Tu Vu, Timo Schick, and Jingbo Shang. 2022. [Leveraging qa datasets to improve generative data augmentation](#). *arXiv preprint arXiv:2205.12604*.
- Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. 2022. [Tuning language models as training data generators for augmentation-enhanced few-shot learning](#). *arXiv preprint arXiv:2211.03044*.
- Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022a. [MetaICL: Learning to learn in context](#). In *NAACL-HLT*.
- So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. 2022b. [FILM: Following Instructions in Language with Modular Methods](#). In *International Conference on Learning Representations (ICLR)*.

- Lester James V Miranda, Yizhong Wang, Yanai Elazar, Sachin Kumar, Valentina Pyatkin, Faeze Brahman, Noah A Smith, Hannaneh Hajishirzi, and Pradeep Dasigi. 2024. Hybrid preferences: Learning to route instances for human vs. ai feedback. *arXiv preprint arXiv:2410.19133*.
- Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2022a. [Reframing instructional prompts to gptk’s language](#). In *Annual Meeting of the Association for Computational Linguistics (ACL) - Findings*.
- Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022b. Cross-Task Generalization via Natural Language Crowdsourcing Instructions. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- MosaicML. 2023. [Introducing mpt-7b: A new standard for open-source, commercially usable llms](#). Blog post.
- Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. [Jfleg: A fluency corpus and benchmark for grammatical error correction](#). In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.
- Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. [The e2e dataset: New challenges for end-to-end generation](#). In *Annual SIGdial Meeting on Discourse and Dialogue*.
- OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training Language Models to Follow Instructions with Human Feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Ellie Pavlick, Matt Post, Ann Irvine, Dmitry Kachaev, and Chris Callison-Burch. 2014. [The Language Demographics of Amazon Mechanical Turk](#). *Transactions of the Association for Computational Linguistics (TACL)*.
- Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. page 12.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *Journal of Machine Learning Research (JMLR)*.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '20*. IEEE Press.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. [Impact of pretraining term frequencies on few-shot reasoning](#). *arXiv preprint arXiv:2202.07206*.

Nils Reimers and Iryna Gurevych. 2019. [Sentence-bert: Sentence embeddings using siamese bert-networks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. [Choice of plausible alternatives: An evaluation of commonsense causal reasoning](#). In *AAAI spring symposium: logical formalizations of commonsense reasoning*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. [WINOGRANDE: an adversarial winograd schema challenge at scale](#). In *Conference on Artificial Intelligence (AAAI)*.

- Marta Sandri, Elisa Leonardelli, Sara Tonelli, and Elisabetta Jezeq. 2023. [Why don't you do it right? analysing annotators' disagreement in subjective tasks](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2428–2441, Dubrovnik, Croatia. Association for Computational Linguistics.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. [Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter](#). In *Advances in Neural Information Processing Systems (NeurIPS) Workshop on Energy Efficient Machine Learning and Cognitive Computing*.
- Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *International Conference on Learning Representations (ICLR)*.
- Timo Schick and Hinrich Schütze. 2021. [Generating datasets with pretrained language models](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Burr Settles. 2009. Active learning literature survey.
- Igor Shalyminov, Alessandro Sordoni, Adam Atkinson, and Hannes Schulz. 2020. [Fast domain adaptation for goal-oriented dialogue using a hybrid generative-retrieval transformer](#). In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
- Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

- Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. [ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks](#). In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Chandan Singh, John X Morris, Jyoti Aneja, Alexander M Rush, and Jianfeng Gao. 2022. [Explaining patterns in data with language models via interpretable autoprompting](#). *arXiv preprint arXiv:2210.01848*.
- Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. 2023. A long way to go: Investigating length correlations in RLHF. *arXiv preprint arXiv:2310.03716*.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, and et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.
- Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. 2020. [Language-Conditioned Imitation Learning for Robot Manipulation Tasks](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. [Revisiting Unreasonable Effectiveness of Data in Deep Learning Era](#). In *International Conference on Computer Vision (ICCV)*.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.
- Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293.
- Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. 2021. Multimodalqa: Complex question answering over text, tables and images. *arXiv preprint arXiv:2104.06039*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. [Stanford alpaca: An instruction-following llama model](#). GitHub repository.

Qwen Team. 2024. [Qwen2.5: A party of foundation models](#).

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023a. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. 2019. [Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples](#). In *International Conference on Learning Representations (ICLR)*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024a. [Large language models are not fair evaluators](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023a. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023b. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. In *Advances in Neural Information Processing Systems*, volume 36, pages 74764–74786.

- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022a. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.
- Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022b. Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ Tasks. In *EMNLP*.
- Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. 2024b. Helpsteer2-preference: Complementing ratings with preferences. *arXiv preprint arXiv:2410.01257*.
- Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. 2024c. Helpsteer2: Open-source dataset for training top-performing reward models. *arXiv preprint arXiv:2406.08673*.
- Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, and Oleksii Kuchaiev. 2023c. HelpSteer: Multi-attribute Helpfulness Dataset for SteerLM. *arXiv preprint arXiv:2311.09528*.
- Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021. [Towards zero-label language learning](#). *arXiv preprint arXiv:2109.09193*.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations (ICLR)*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.
- Nathaniel Weir, Xingdi Yuan, Marc-Alexandre Côté, Matthew Hausknecht, Romain Laroche, Ida Momennejad, Harm Van Seijen, and Benjamin Van Durme. 2022. [One-Shot Learning from a Demonstration with Hierarchical Latent Language](#). *arXiv preprint arXiv:2203.04806*.

- Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. [Generating sequences by learning to self-correct](#). *arXiv preprint arXiv:2211.00053*.
- Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew Peters. 2020. Learning from Task Descriptions. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Peter West, Chandra Bhagavatula, Jack Hessel, Jena D Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2021. [Symbolic knowledge distillation: from general language models to commonsense models](#). In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [Transformers: State-of-the-Art Natural Language Processing](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP) - System Demonstrations*.
- Zequi Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2024. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems*, 36.
- Mengzhou Xia, Antonios Anastasopoulos, Ruo Chen Xu, Yiming Yang, and Graham Neubig. 2020. Predicting performance for natural language processing tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8625–8646.
- Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*.
- Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020. [Self-training with noisy student improves imagenet classification](#). In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10687–10698.
- Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le,

- Tengyu Ma, and Adams Wei Yu. 2024. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36.
- Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. [UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models](#). *arXiv preprint arXiv:2201.05966*.
- Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*.
- Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. 2022. [Zero-Prompt: Scaling Prompt-Based Pretraining to 1,000 Tasks Improves Zero-Shot Generalization](#). *arXiv preprint arXiv:2201.06910*.
- Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. [mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer](#). In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.
- Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. [Generative data augmentation for commonsense reasoning](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP) - Findings*.

- Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021a. [CrossFit: A Few-shot Learning Challenge for Cross-task Generalization in NLP](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Qinyuan Ye and Xiang Ren. 2021. [Learning to Generate Task-Specific Adapters from Task Description](#). In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Seonghyeon Ye, Doyoung Kim, Joel Jang, Joongbo Shin, and Minjoon Seo. 2022. [Guess the instruction! making language models stronger zero-shot learners](#). *arXiv preprint arXiv:2210.02969*.
- Zihuiwen Ye, Pengfei Liu, Jinlan Fu, and Graham Neubig. 2021b. Towards more fine-grained and reliable nlp performance prediction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3703–3714.
- Eric Zelikman, Jesse Mu, Noah D Goodman, and Yuhuai Tony Wu. 2022. [STar: Self-taught reasoner bootstrapping reasoning with reasoning](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. [Opt: Open pre-trained transformer language models](#). *arXiv preprint arXiv:2205.01068*.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. [Bertscore: Evaluating text generation with bert](#). In *International Conference on Learning Representations (ICLR)*.
- Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. [\(InThe\)WildChat: 570K ChatGPT Interaction Logs In The Wild](#). In *The Twelfth International Conference on Learning Representations*.
- Xuandong Zhao, Siqi Ouyang, Zhiguo Yu, Ming Wu, and Lei Li. 2022. [Pre-trained language models can be fully zero-shot learners](#). *arXiv preprint arXiv:2212.06950*.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023a. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623.

- Lianmin Zheng, Ying Sheng, Wei-Lin Chiang, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023b. [Chatbot Arena: Benchmarking LLMs in the Wild with Elo Ratings](#). Blog post.
- Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021. [Adapting Language Models for Zero-shot Learning by Meta-tuning on Dataset and Prompt Collections](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP) - Findings*.
- Chunting Zhou, Junxian He, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022a. [Prompt Consistency for Zero-Shot Task Generalization](#). In *Conference on Empirical Methods in Natural Language Processing (EMNLP) - Findings*.
- Chunting Zhou, Pengfei Liu, Puxin Xu, Srinii Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023a. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.
- Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023b. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.
- Yongchao Zhou, Andrei Ioan Muresanu, Ziwon Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022b. [Large language models are human-level prompt engineers](#). *arXiv preprint arXiv:2211.01910*.

Appendix A

Appendix: Unifying NLP Tasks by Adding Instructions

A.1 Crowdsourcing Human Feedback

We use Amazon Mechanical Turk (AMT) to crowdsource feedback on the quality of the collected instructions. We limit our crowdworkers to predominantly English-speaking countries (USA, UK, Canada, and Australia), and to those who have finished over $1k$ HITs with an approval rating of over 99%.

Fig. A.1 shows the crowdsourcing template used for collecting crowdworker feedback on our instructions. We show the instructions (the task definition, along with positive and negative examples) followed by forms for their feedback. We allow the crowdworkers to give us a qualitative measure of their perceived quality as well as text boxes for more concrete items (such as typos or phrasings that may benefit from more clear articulation). For each task, we solicit the feedback of 3 crowdworkers and then use this feedback to improve the task definitions or the examples for each task.

A.2 Crowdsourcing Human Judgements of Generation Quality

We perform a crowdsourcing experiment on Amazon Mechanical Turk (AMT) to assess the quality of the generated responses of models. Specifically, we ask crowdworkers to indicate if they prefer the predicted answer by the model or the ground truth outputs for each instances. The annotation interface is shown in

Instructions and Examples	Your Task
<p>The main purpose of this task is to get your feedback on the quality of the provided instructions below. We use this feedback to improve the quality of the provided instructions for the next round. Please read the instructions and tell us what you think. You can tell us about: (1) typos (2) ambiguous or odd phrasings (3) what can be added/changed to make the task easier to understand. Feel free to be as critical as you want about the instructions.</p> <p>Instructions: Given a statement and question, generate the answer to the question such that the answer is contained in the statement.</p> <p>Click here to hide/show positive examples</p> <p>Positive (Desirable) Examples</p> <p>Here are examples of desirable behaviors for this task:</p> <hr/> <p>Example 1: Input: statement: Two lenses to produce magnified images allow for detailed observation of stars., question: Two lenses to produce magnified images allow for what? Output: Detailed observation of stars Explanation of the example: The answer generated is correct with respect to the question and the statement</p> <hr/> <p>Example 2: Input: statement: Neurons deliver electrical impulses to other cells, question: What do neurons deliver to other cells? Output: Electrical impulses Explanation of the example: The answer generated is correct with respect to the question and the statement</p> <hr/> <p>Click here to hide/show negative examples</p> <p>Negative (Undesirable) Examples</p> <p>Here are several negative examples (undesirable outputs) for this task:</p> <hr/> <p>Example 1: Input: statement: Sharks have sharp jaws to catch their prey, question: What do sharks have to catch their prey? Output: Sharp claws Explanation of the example: The answer generated is incorrect with respect to the question and the statement</p> <hr/> <p>Example 2: Input: statement: Two lenses to produce magnified images allow for detailed observation of stars., question: Two lenses to produce magnified images allow for what? Output: Barack Obama Explanation of the example: The example is incorrect, as the answer generated is not related to the statement or question</p> <hr/> <p>Great! You are now ready to start the following examples!</p>	<p>First, please tell us how we can improve the above instructions. We expect you to provide suggestions to improve the instructions for every few HITs. So, please do not leave them empty all the time. If the instructions are impeccable, leave the boxes empty (and avoid unconstructive feedback such as "it's poor", "nothing", etc.)</p> <p>Instruction quality: Q1: What did you think about the quality of the task definition? (Avoid putting random/garbage text here. Consistent random responses will result in being blocked. Better leave this box empty instead of writing something irrelevant. We bonus those workers with constructive feedback for improving the instructions.) <input type="radio"/> pretty good <input type="radio"/> okay <input type="radio"/> weak <input type="radio"/> confusing</p> <p>Q2: What can be improved in the instructions, if any? (please indicate any typos, lack of clarities or any other issues)</p> <div style="border: 1px solid black; height: 30px; width: 100%;"></div> <hr/> <p>Positive examples quality: Q1: What did you think about the quality of the provided positive examples? <input type="radio"/> pretty good <input type="radio"/> okay <input type="radio"/> weak <input type="radio"/> confusing</p> <p>Q2: What can be improved about the positive examples, if any? (please indicate any typos, lack of clarities or any other issues)</p> <div style="border: 1px solid black; height: 30px; width: 100%;"></div> <hr/> <p>Negative examples quality: Q1: What did you think about the quality of the provided negative examples? <input type="radio"/> pretty good <input type="radio"/> okay <input type="radio"/> weak <input type="radio"/> confusing</p> <p>Q2: What can be improved about the negative examples, if any? (please indicate any typos, lack of clarities or any other issues)</p> <div style="border: 1px solid black; height: 30px; width: 100%;"></div> <hr/> <p style="color: red; text-align: center;">Now please answer the following inputs, according to the provided instructions above.</p> <p>Input: statement: Mice catch small creatures to eat, question: What species catch small creatures to eat?</p> <p>Now, given the above input, write an appropriate output:</p> <div style="border: 1px solid black; height: 30px; width: 100%;"></div> <hr/> <p>Input: statement: a cat requires vision for seeing food animals, question: What does a cat require vision for seeing?</p>

Figure A.1: The crowdsourcing template we use to receive feedback on our collected tasks.

Fig. A.2 which is essentially the same template used for the quality assessment of the dataset (§A.1), except that here the crowdworkers are shown a *pair* of responses for each instances—the reference text (from our benchmark) and the one generated by the model—turning the task into a *comparative* evaluation.

For each instance, we obtain annotations from an annotators as to whether they prefer either responses over the other or they would rate them equally (“tie”). The model receives a credit of 1.0 if the worker favors the model’s prediction at least as good as the ground truth label (otherwise, the model would receive a credit of 0.0). The overall accuracy score for the model is computed by averaging instance-level scores. To reduce the costs, the human evaluation of our models is done on 60 randomly selected tasks (about half of our evaluation tasks), and on 10 random instances of each task.

Since it is non-trivial to find non-English speaking crowdworkers [Pavlick et al., 2014], this evaluation was restricted to English language tasks. Therefore, since our task is focused on English tasks, we required workers to be based in a country with a population predominantly of native English speakers (e.g., USA, Canada, UK, and Australia) and have completed at least 5000 HITs with $\geq 99\%$ assignment approval rate.

The resulting human-evaluation metric indicates how often were model predictions at least as good as our

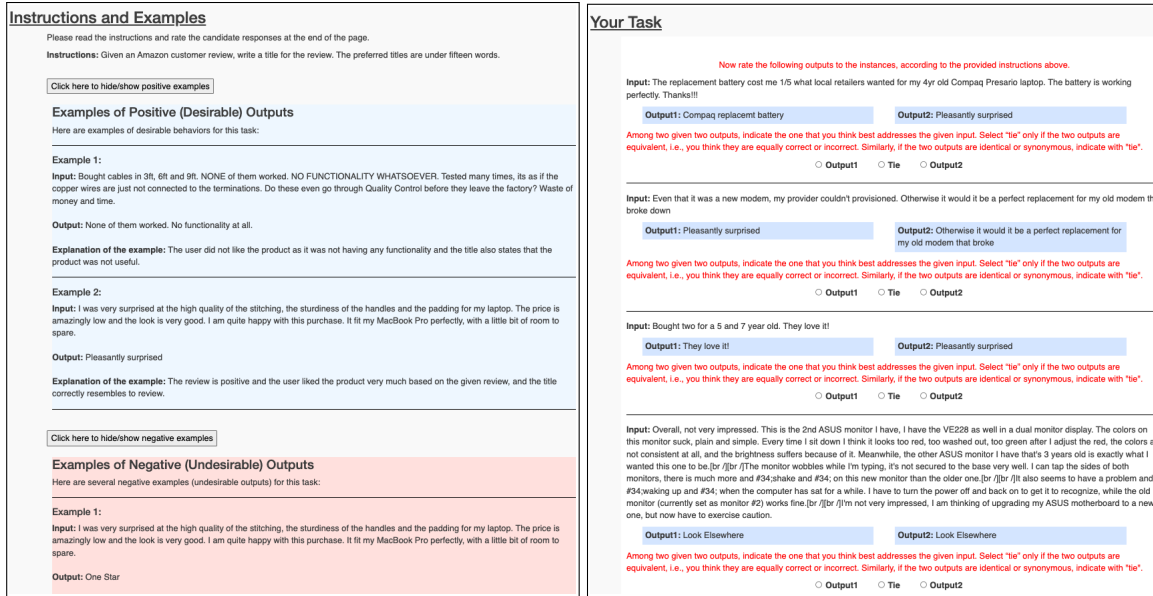


Figure A.2: Crowdsourcing interface used for human assessment of our baselines (§5.4.3).

ground truth labels. In this evaluation, the theoretical upperbound is 100% where the model is rated at least as good as the ground truth. The results of human evaluation are shown in the bottom row of Fig. 2.5.

A.3 Instruction Schema

We instruction schema is based on that of NATINST [Mishra et al., 2022b] but we simply it largely for make data collection easier. Our DEFINITION field serves as the union of Mishra et al. [2022b]’s DEFINITION, THINGS TO AVOID, and EMPHASIS & CAUTION. Additionally, we drop their TITLE and PROMPT as their content is most often covered by DEFINITION.

A.4 Model Implementation Details

T5 experiments. We use T5 for training our T_k -INSTRUCT, estimating the performance of supervised approach, and conducting analysis. Our experiments that finetune the T5-11B model are conducted based on the Google’s T5 library¹ and we use their T5.1.1.xml checkpoint² by default, which is pre-trained only on C4.

¹<https://github.com/google-research/text-to-text-transfer-transformer>

²https://console.cloud.google.com/storage/browser/t5-data/pretrained_models/t5.1.1.xml

These experiments are run on Google V3-256 TPUs using a batch size of 1,048,576 tokens (1,024 examples), a constant learning rate of $1e-5$ and a total of 1000 steps. Each training takes 4 hours to finish.

Our analyses that use T5 models smaller than 11B are conducted based on Huggingface’s transformers library and model checkpoints³ [Wolf et al., 2020] on GPU machines. When fine-tuning models, we train them for two epochs with a batch size of 16 and a constant learning rate of $1e-5$. The maximum input length is set to 1024, and the maximum output length is set to 128. These experiments are conducted with 8 A100 GPUs with 48GB GPU memory per each. We use DeepSpeed⁴ for model parallelization, with bfloat16 precision enabled to save the GPU memory. Each training takes 6 hours to finish.

GPT3 and InstructGPT experiments. We use the OpenAI API⁵ for conducting the GPT3 experiments. We use their “davinci” engine for the GPT3 language model experiments and their “text-davinci-001” engine for the InstructGPT experiments. When making the requests, we set the temperature as 0, top_p as 1 and the maximum generation length as 128. Due to the expensive cost, we randomly sample 20 instances from each of our 119 test tasks to estimate the performance of GPT3 and InstructGPT. All API requests were made on May 30th, 2022.

Encoding instruction with input For every problem setup, we map a given instruction I_t and an input instance x into a textual format obtain $enc(I_t, x)$. Each instruction I_t consists of multiple elements as described in our instruction schema (§2.3). We map each element of the instruction to a textual format and append it before the input instance. Fig. A.3 shows how we encode the full instruction. We study different combinations of these instruction elements in §2.6.2. The encoded instance is then fed to an encoder-decoder model to predict y : $M : enc(I_t, x) \rightarrow y$.

A.5 Evaluation Metrics

We adopt ROUGE-L as our automatic evaluation metric in this work. However, it remains a question how much ROUGE-L can reflect model’s performance on different tasks. Although we cannot test ROUGE-

³<https://huggingface.co/docs/transformers/index>

⁴<https://github.com/microsoft/DeepSpeed>

⁵<https://beta.openai.com/docs/introduction/overview>

```

Definition : {{definition}}
Positive Example 1–
  input : {{p_ex1.input}}
  output : {{p_ex1.output}}
  explanation : {{p_ex1.exp}}
Positive Example 2–
  ...
Negative Example 1–
  input : {{n_ex1.input}}
  output : {{n_ex1.output}}
  explanation : {{n_ex1.exp}}
Negative Example 2–
  ...
Now complete the following example–
  input : {{x.input}}
  output :

```

Figure A.3: Encoding task instruction with input.

L’s correlation with each task-specific metric of the tasks included in our data, we do investigate whether ROUGE-L can be used for classification tasks. Fig. A.4 plots the ROUGE-L scores and accuracy of several models on different types of tasks. These task types are usually regarded as classification tasks, and have very short ground truth output. We can see that for all these task types, the trend of ROUGE-L correlates well with the trend of accuracy. For some task types, we do see some gap between these two metrics. The reason is because there are some generation tasks categorized into these types. These results indicate that ROUGE-L is a good proxy for accuracy for classification tasks.

A.6 Distribution of Tasks

As is described in §2.3, SUPERNI provides the annotation for categorizing tasks along three different dimensions: task type, language and domain. Fig. A.5 shows the distribution of tasks among these three dimensions. This meta-information can be used to study model’s generalization ability in different senses. Despite the diversity of the data, we acknowledge the skew toward certain tasks and languages, which we leave to be addressed by future work.

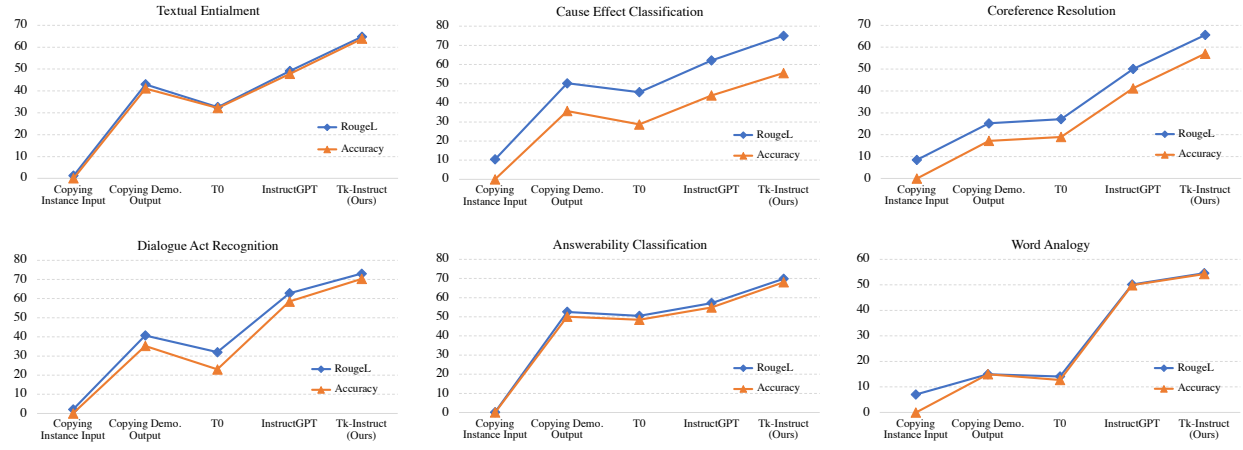


Figure A.4: ROUGE-L v.s. Accuracy for task types that are usually regarded as classification tasks. The trends of these two metrics are highly correlated with each other even if not overlapped.

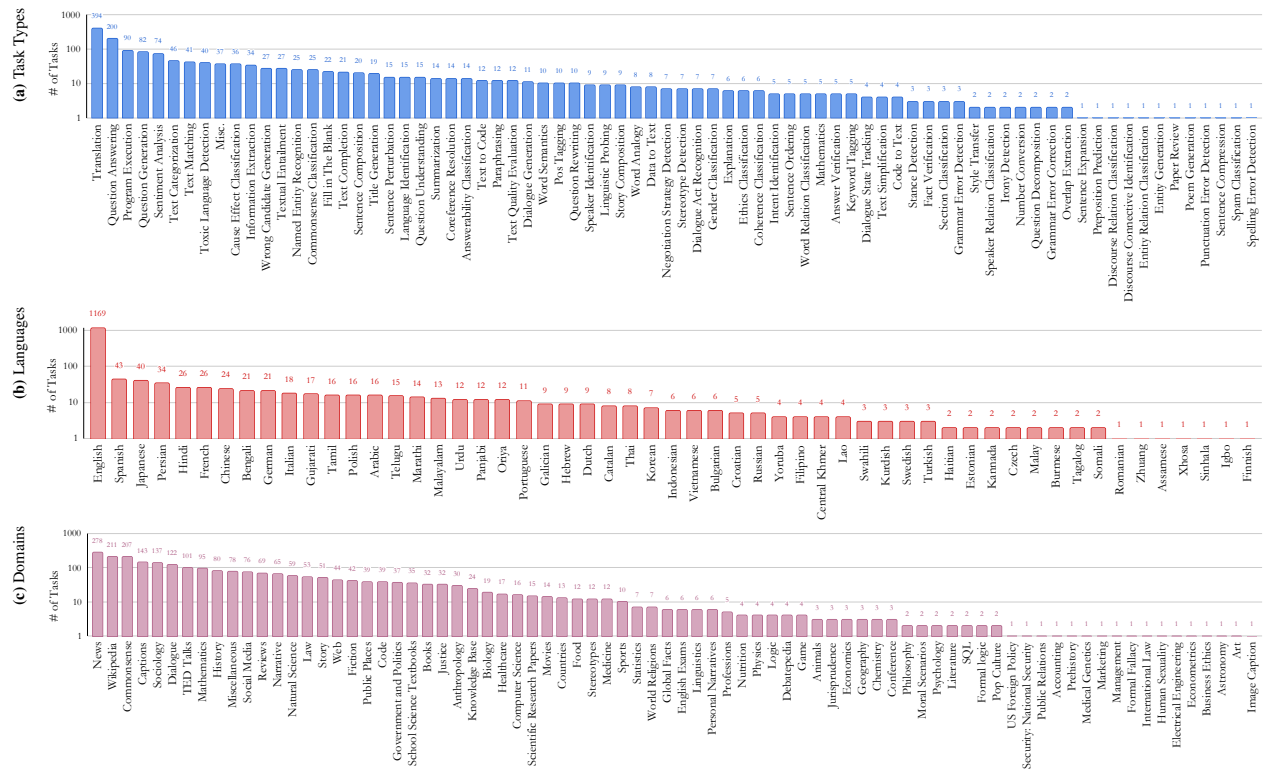


Figure A.5: Distribution of SUPERNI tasks in terms of their (a) task types (b) languages (c) domains. y -axes are in log scale.

A.7 Evaluation Tasks

Table A.1 lists the 12 task categories used for our evaluation and all the tasks included in each category (introduced in §2.5.1). To provide a better sense of what those tasks look like, we also select one representative task from each category and list them in Table A.2 - A.13. Due to the large number of tasks in our dataset, we cannot list all 1,616 tasks in this paper. We refer the reader to our dataset.

A.8 Performance Improvement per Evaluation Task

To provide more detailed analysis of *Tk*-INSTRUCT on each individual task, Fig. A.6 presents the per-task improvement of our *Tk*-INSTRUCT (3B) model over the best of two heuristic baselines on the English evaluation tasks, and Fig. A.7 presents the per-task improvement of the *mTk*-INSTRUCT model on the cross-lingual evaluation tasks. For most of the evaluation tasks, we see a significant extent of generalization by *Tk*-INSTRUCT.

Task Category	Metric	List of Tasks
Textual Entailment	Exact Match	task937_defeasible_nli_atomic_textual_entailment task202_multinli_textual_entailment task936_defeasible_nli_atomic_textual_entailment task641_e_snl_textual_entailment task1344_rte_textual_entailment task1615_sick_textual_entailment task1385_anli_textual_entailment task935_defeasible_nli_atomic_textual_entailment task199_multinli_textual_entailment task1388_cb_textual_entailment task1554_scitail_textual_entailment task640_e_snl_textual_entailment task534_farstail_textual_entailment task201_multinli_textual_entailment task1386_anli_textual_entailment task463_pasinlu_textual_entailment task1387_anli_textual_entailment task738_spectrum_textual_entailment task1529_scitailv1.1_textual_entailment task190_snl_textual_entailment task200_multinli_textual_entailment task1612_sick_textual_entailment task970_sherlic_textual_entailment task890_gwsd_textual_entailment task464_pasinlu_textual_entailment task1516_imppres_textual_entailment task642_e_snl_textual_entailment
Cause Effect Classification	Exact Match	task1178_xcopa_cause_effect_classification task391_cod3s_cause_effect_classification task939_indicnlp_cause_effect_classification task392_cod3s_cause_effect_classification task938_indicnlp_cause_effect_classification task1168_xcopa_cause_effect_classification task828_copa_cause_effect_classification task1628_copa_hr_cause_effect_classification task943_indicnlp_cause_effect_classification task1182_xcopa_cause_effect_classification task1171_xcopa_cause_effect_classification task968_xcopa_cause_effect_classification task942_indicnlp_cause_effect_classification task1181_xcopa_cause_effect_classification task1172_xcopa_cause_effect_classification task1393_copa_cause_effect_classification task1174_xcopa_cause_effect_classification task1627_copa_hr_cause_effect_classification task1177_xcopa_cause_effect_classification task1184_xcopa_cause_effect_classification task1185_xcopa_cause_effect_classification task1176_xcopa_cause_effect_classification task614_glucose_cause_effect_classification task1629_copa_hr_cause_effect_classification task1175_xcopa_cause_effect_classification task827_copa_cause_effect_classification task1173_xcopa_cause_effect_classification task1180_xcopa_cause_effect_classification task1170_xcopa_cause_effect_classification task1183_xcopa_cause_effect_classification task969_xcopa_cause_effect_classification task941_indicnlp_cause_effect_classification task1626_copa_hr_cause_effect_classification task940_indicnlp_cause_effect_classification task393_cod3s_cause_effect_classification task1169_xcopa_cause_effect_classification task1179_xcopa_cause_effect_classification
Coreference Resolution	Exact Match	task1391_winogrande_coreference_resolution task1664_wino_bias_coreference_resolution task304_numeric_fused_head_coreference_resolution task892_gap_coreference_resolution task891_gap_coreference_resolution task330_gap_coreference_resolution task401_numeric_fused_head_coreference_resolution task879_schema_guided_dstc8_dialogue_act_recognition task362_spolin_dialogue_act_recognition task1533_dailydialog_dialogue_act_recognition task1534_dailydialog_dialogue_act_recognition task033_winogrande_coreference_resolution task133_winowhy_coreference_resolution task329_gap_coreference_resolution task249_enhanced_wsc_coreference_resolution task648_winograd_wsc_coreference_resolution task1390_wsc_fixed_coreference_resolution task893_gap_coreference_resolution
Dialogue Act Recognition	Exact Match	task879_schema_guided_dstc8_dialogue_act_recognition task362_spolin_dialogue_act_recognition task1533_dailydialog_dialogue_act_recognition task1534_dailydialog_dialogue_act_recognition task880_schema_guided_dstc8_dialogue_act_recognition task1531_dailydialog_dialogue_act_recognition task1394_meta_woz_dialogue_act_recognition
Answerability Classification	Exact Match	task020_mctaco_answerability_classification task050_multire_answerability_classification task1439_doqa_answerability_classification task233_iirc_answerability_classification task226_stack_overflow_answerability_classification task396_persianqa_answerability_classification task1640_adverserial_qa_answerability_classification task232_iirc_answerability_classification task1442_doqa_answerability_classification task242_tweetqa_answerability_classification task1624_disfl_qa_answerability_classification task520_aquamuse_answerability_classification task290_tellmewhy_answerability_classification task349_squad2.0_answerability_classification
Word Analogy	Exact Match	task1155_bard_word_analogy task1152_bard_word_analogy task1158_bard_word_analogy task1156_bard_word_analogy task1157_bard_word_analogy task1159_bard_word_analogy task1153_bard_word_analogy task1154_bard_word_analogy
Overlap Extraction	ROUGE-L	task039_qasc_overlap_extraction task281_points_of_correspondence_overlap_extraction
Keyword Tagging	ROUGE-L	task613_liar_keyword_tagging task645_wiki_auto_all_data_keyword_tagging task620_ohsumed_keyword_tagging task036_qasc_keyword_tagging task623_ohsumed_keyword_tagging
Question Rewriting	ROUGE-L	task670_ambigqa_question_rewriting task121_zest_question_rewriting task1195_disfl_qa_question_rewriting task442_com_qa_question_rewriting task1345_qqp_question_rewriting task035_winogrande_question_rewriting task671_ambigqa_question_rewriting task1562_zest_question_rewriting task1622_disfl_qa_question_rewriting task034_winogrande_question_rewriting task402_grailqa_question_rewriting
Title Generation	ROUGE-L	task1356_xlsum_title_generation task1540_peer_read_title_generation task1659_billsum_title_generation task569_recipe_nlg_title_generation task1342_amazon_us_reviews_title_generation task220_rocstories_title_generation task1561_clickbait_news_bg_title_generation task418_persent_title_generation task1358_xlsum_title_generation task769_qed_title_generation task219_rocstories_title_generation task602_wikitext_title_generation task1586_scifact_title_generation task743_eurlex_title_generation task500_scruples_title_generation task619_ohsumed_title_generation task510_reddit_tifu_dataset_title_generation task288_gigaword_title_generation task1161_coda_19_title_generation
Data to Text	ROUGE-L	task957_e2e_data_to_text task1631_open_pi_data_to_text task1598_nyc_data_to_text task1728_web_nlg_data_to_text task102_commongen_data_to_text task677_ollie_data_to_text task1407_dart_data_to_text task1409_dart_data_to_text task760_msr_sqa_data_to_text
Grammar Error Correction	ROUGE-L	task1557_jfleg_grammar_error_correction

Table A.1: 12 Evaluation categories (§2.5.1), their evaluation metrics (Exact Matching or ROUGE-L, §2.5.1), and all the tasks in each category.

Task Type	Textual Entailment
Task ID	task1344_rte_textual_entailment
Definition	In this task, you're given two sentences. Indicate if the first sentence clearly entails the second sentence (i.e., one can conclude the 2nd sentence by reading the 1st one). Indicate your answer with "1" if the first sentence entails the second sentence, otherwise answer with "0".
Positive Ex-ample	Input: Sentence 1: No Weapons of Mass Destruction Found in Iraq Yet. Sentence 2: Weapons of Mass Destruction Found in Iraq. Output: 0 Explanation: In our first statement we clearly say that Iraq does not have any weapon of mass destruction but the second sentence says that weapon of mass destruction is found in Iraq which is a contradiction. Hence output will be 0 for non entailment.
Negative Example	Input: Sentence 1: Valero Energy Corp., on Monday, said it found "extensive" additional damage at its 250,000-barrel-per-day Port Arthur refinery. Sentence 2: Valero Energy Corp. produces 250,000 barrels per day. Output: 0 Explanation: The first statement mentions that there was damage found in the 250,000 barrel-per-day Port Arthur refinery. Which means that they produce 250,000 barrels a day. Hence the output should have been 1 for entailment.
Instance	Input: Sentence 1: Like the United States, U.N. officials are also dismayed that Aristide killed a conference called by Prime Minister Robert Malval in Port-au-Prince in hopes of bringing all the feuding parties together. Sentence 2: Aristide had Prime Minister Robert Malval murdered in Port-au-Prince. Valid Output: ["0"]

Table A.2: An example task in the Textual Entailment category of our dataset, adopted from RTE Dagan et al. [2005]; Bentivogli et al. [2008].

Task Type	Cause Effect Classification
Task ID	task828_copa_cause_effect_classification
Definition	In this task your given two statements. You must judge whether the second sentence is the cause or effect of the first one. Label the instances as "cause" or "effect" based on your judgment. The sentences are separated by a newline character.
Positive Ex-ample	Input: The women met for coffee. They wanted to catch up with each other. Output: cause Explanation: The women met for coffee because they wanted to catch up with each other.
Negative Example	Input: My body cast a shadow over the grass. The sun was rising. Output: effect Explanation: The rising of the sun isn't an effect of casting a shadow over the grass.
Instance	Input: The woman tolerated her friend's difficult behavior. The woman knew her friend was going through a hard time. Valid Output: ["cause"]

Table A.3: An example task in the Cause Effect Classification category of our dataset, adopted from COPA Roemmele et al. [2011].

Task Type	Coreference Resolution
Task ID	task1391_winogrande_coreference_resolution
Definition	In this task, you are given a question containing a blank () and two options. You should pick the best option to answer the question. Please answer with "A" or "B".
Positive Example	Input: Katrina gave Christine a stuffed animal for their birthday, but _ already had this one. (A) Katrina (B) Christine Output: B Explanation: Since the blank is someone who received the gift and already had a stuffed animal, the answer must be "Christine".
Negative Example	Input: Kevin had to use less sunscreen when at the beach tanning than Justin because _ had less sensitive skin. (A) Kevin (B) Justin Output: (A) Explanation: Here, an additional parentheses has been added to the answer. Note that, a valid answer must be "A" or "B".
Instance	Input: Benjamin hated being in the sand and just watched Nelson make castle since _ hated to be messy. (A) Benjamin (B) Nelson Valid Output: ["A"]

Table A.4: An example task in the Cause Effect Classification category of our dataset, adopted from WinoGrande Sakaguchi et al. [2020].

Task Type	Dialogue Act Recognition
Task ID	task1394_meta_woz_dialogue_act_recognition
Definition	In this task, you are given four sentences: a bot task sentence, a bot role sentence, a user task sentence and a user role sentence. Your job is to classify given sentences into one of the 47 different domains. The domains are: "UPDATE_CALENDAR", "PRESENT_IDEAS", "MOVIE_LISTINGS", "AUTO_SORT", "GAME_RULES", "CONTACT_MANAGER", "BANK_BOT", "MUSIC_SUGGESTER", "CHECK_STATUS", "PET_ADVICE", "HOW_TO_BASIC", "NAME_SUGGESTER", "QUOTE_OF_THE_DAY_BOT", "GUINNESS_CHECK", "INSURANCE", "RESTAURANT_PICKER", "MAKE_RESTAURANT_RESERVATIONS", "WEDDING_PLANNER", "SKI_BOT", "HOME_BOT", "PLAY_TIMES", "BUS_SCHEDULE_BOT", "WHAT_IS_IT", "PHONE_PLAN_BOT", "DECIDER_BOT", "PHONE_SETTINGS", "TIME_ZONE", "LIBRARY_REQUEST", "UPDATE_CONTACT", "CATALOGUE_BOT", "PROMPT_GENERATOR", "SCAM_LOOKUP", "SPORTS_INFO", "POLICY_BOT", "CITY_INFO", "APARTMENT_FINDER", "EVENT_RESERVE", "SHOPPING", "EDIT_PLAYLIST", "LOOK_UP_INFO", "ORDER_PIZZA", "WEATHER_CHECK", "APPOINTMENT_REMINDER", "GEOGRAPHY", "STORE_DETAILS", "AGREEMENT_BOT", "ALARM_SET".
Positive Example	Input: Bot's task: Inform the user that the topping they are asking for is unavailable. Bot's role: You are a bot designed to help customers order pizza. User's task: Ask if a certain pizza topping is available. User's role: You are interacting with a pizza restaurant bot. Output: ORDER_PIZZA Explanation: According to the descriptions of the four sentences, we know that the type of task is ORDER_PIZZA.
Negative Example	Input: Bot's task: Help the user with their pizza order. Bot's role: You are a bot designed to help customers order pizza. User's task: Ask the bot for three different pizzas. User's role: You are interacting with a pizza restaurant bot. Output: UPDATE_CALENDAR Explanation: According to the descriptions of the tasks and roles, we know that the type of task is ORDER_PIZZA, but the output is UPDATE_CALENDAR, so it is incorrect.
Instance	Input: Bot's task: Tell the user when the movie is playing on Friday night. Bot's role: You are a bot designed to provide movie listings. User's task: Ask the bot for the movie times for a movie on Friday night. User's role: You are interacting with a bot designed to provide movie listings. Valid Output: ["MOVIE_LISTINGS"]

Table A.5: An example task in the Dialogue Act Recognition category of our dataset, adopted from MetaLWOz Shalymov et al. [2020].

Task Type	Answerability Classification
Task ID	task1640_adversarial_qa_answerability_classification
Definition	Given a paragraph from a wikipedia article about some topic, and a question related to the topic, determine whether the question is answerable from the paragraph. If the question is answerable, answer “True”, otherwise, answer “False”.
Positive Example	<p>Input: Another approach to brain function is to examine the consequences of damage to specific brain areas. Even though it is protected by the skull and meninges, surrounded by cerebrospinal fluid, and isolated from the bloodstream by the blood 2013 brain barrier, the delicate nature of the brain makes it vulnerable to numerous diseases and several types of damage. In humans, the effects of strokes and other types of brain damage have been a key source of information about brain function. Because there is no ability to experimentally control the nature of the damage, however, this information is often difficult to interpret. In animal studies, most commonly involving rats, it is possible to use electrodes or locally injected chemicals to produce precise patterns of damage and then examine the consequences for behavior. Question: What is surrounded by cerebrospinal fluid?</p> <p>Output: True</p> <p>Explanation: The paragraph comes from the wikipedia page on the brain. The answer to the question is the brain which can be found in the paragraph.</p>
Negative Example	<p>Input: NASCAR (headquartered in Daytona Beach) begins all three of its major auto racing series in Florida at Daytona International Speedway in February, featuring the Daytona 500, and ends all three Series in November at Homestead-Miami Speedway. Daytona also has the Coke Zero 400 NASCAR race weekend around Independence Day in July. The 24 Hours of Daytona is one of the world’s most prestigious endurance auto races. The Grand Prix of St. Petersburg and Grand Prix of Miami have held IndyCar races as well. Question: What is the starting time of NASCAR’s big events?</p> <p>Output: False</p> <p>Explanation: This paragraph comes from the wikipedia article on Florida. The answer to the given question is February which can be found in the paragraph, however the output is given as False.</p>
Instance	<p>Input: Another approach to brain function is to examine the consequences of damage to specific brain areas. Even though it is protected by the skull and meninges, surrounded by cerebrospinal fluid, and isolated from the bloodstream by the blood 2013 brain barrier, the delicate nature of the brain makes it vulnerable to numerous diseases and several types of damage. In humans, the effects of strokes and other types of brain damage have been a key source of information about brain function. Because there is no ability to experimentally control the nature of the damage, however, this information is often difficult to interpret. In animal studies, most commonly involving rats, it is possible to use electrodes or locally injected chemicals to produce precise patterns of damage and then examine the consequences for behavior. Question: What are the benefits of the blood brain barrier?</p> <p>Valid Output: [“True”]</p>

Table A.6: An example task in the Answerability Classification category of our dataset, adopted from AdversarialQA Bartolo et al. [2020].

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	<p>Input: eat : fork. cook : ?</p> <p>Output: pan</p> <p>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.</p>
Negative Example	<p>Input: dig : shovel. wash : ?</p> <p>Output: sink</p> <p>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.</p>
Instance	<p>Input: cut : knife. wash : ?</p> <p>Valid Output: [“soap”, “washcloth”, “detergent”, “rag”]</p>

Table A.7: An example task in the Word Analogy category of our dataset, adopted from BARD Fulda et al. [2017].

Task Type	Overlap Extraction
Task ID	task281_points_of_correspondence_overlap_extraction
Definition	You will be given three sentences. Read them, then identify a noun phrase (person, place, or thing) or event that is shared between all three sentences. As the output, write the span of the text corresponding to that phrase in each sentence. Keep the order of the sentences, that is, your answer should look like: 1: *a phrase from sentence 1* 2: *a phrase from sentence 2* 3: *a phrase from sentence 3*
Positive Example	<p>Input: 1: Four employees of the store have been arrested , but its manager – herself a woman – was still at large Saturday , said Goa police superintendent Kartik Kashyap . 2: If convicted , they could spend up to three years in jail , Kashyap said . 3: The four store workers arrested could spend 3 years each in prison if convicted .</p> <p>Output: 1: Four employees of the store 2: they 3: The four store workers</p> <p>Explanation: All three mentioned parts refer to the same entity, the four employees. “of the store” in first sentence must be included, since it is part of the same noun phrase describing the employees.</p>
Negative Example	<p>Input: 1: But an Arizona official told CNN Bates never trained with the agency . 2: He didn’t come to Arizona , the official from the Maricopa County Sheriff ’s Office said , and he certainly didn’t train with us . 3: Maricopa County Sheriff ’s Office in Arizona says Robert Bates never trained with them .</p> <p>Output: 1: the agency 3: Maricopa County Sheriff ’s Office in Arizona</p> <p>Explanation: The two noun phrases given in this example are correct, but there’s no noun phrase from sentence 2. You should include all three sentences in your response.</p>
Instance	<p>Input: 1: The President is headed to Panama for a regional summit , and Julie Pace of The Associated Press reports one of the big questions is whether he ’ll make history and have a face-to-face meeting with Cuban leader Raul Castro . 2: And so what the White House is going to be weighing is whether this meeting would be a way to generate more progress or whether it would be a premature reward for the Castros . 3: White House weighing whether Obama should meet with Raul Castro .</p> <p>Valid Output: [“1: Cuban leader Raul Castro 2: the Castros 3: Raul Castro”, “1: face-to-face meeting 2: this meeting 3: meet”]</p>

Table A.8: An example task in the Overlap Extraction category of our dataset, adopted from PointsOfCorrespondence [Lebanoff et al. \[2020\]](#).

Task Type	Keyword Tagging
Task ID	task620_ohsumed_keyword_tagging
Definition	Given an abstract, generate a keyword (a noun phrase) that best describes the focus or contribution of the paper. Such keywords can be directly from the given abstract or outside it.
Positive Example	<p>Input: Abstract: Our results suggest that ethylene oxide retention after sterilization is increased in cuprammonium cellulose plate dialyzers containing potting compound. In contrast, cuprammonium cellulose plate dialyzers without potting compound were characterized by a rapid disappearance of retained ethylene oxide after sterilization. Whether these findings explain the low incidence of SARD with cuprammonium cellulose plate dialyzers that do not contain potting material is a matter for continued study and experimentation.</p> <p>Output: Sterilization</p> <p>Explanation: This term is directly present in the abstract and it is one of the main topic in it. So can be chosen as the medical subject heading.</p>
Negative Example	<p>Input: Abstract: Our results suggest that ethylene oxide retention after sterilization is increased in cuprammonium cellulose plate dialyzers containing potting compound. In contrast, cuprammonium cellulose plate dialyzers without potting compound were characterized by a rapid disappearance of retained ethylene oxide after sterilization. Whether these findings explain the low incidence of SARD with cuprammonium cellulose plate dialyzers that do not contain potting material is a matter for continued study and experimentation.</p> <p>Output: Plasma Volume</p> <p>Explanation: This term is not directly present in the abstract and it is no way related to the abstract. So can not be chosen as the medical subject heading. "Cellulose" can be become a mesh term</p>
Instance	<p>Input: Abstract: There is controversy regarding the appropriate utilization of health care resources in the management of tricyclic antidepressant overdose. Antidepressant overdose patients presenting to the emergency department (ED) are routinely admitted to intensive care units, but only a small proportion develop cardiac arrhythmias or other complications requiring such an environment. The authors reviewed the findings in 165 patients presenting to an ED with antidepressant overdose. They found that major manifestations of toxicity on ED evaluation (altered mental status, seizures, arrhythmias, and d conduction defects) were commonly associated with a complicated hospital course. Patients with the isolated findings of sinus tachycardia or QTc prolongation had no complications. No patient experienced a serious toxic event without major evidence of toxicity on ED evaluation and continued evidence of toxicity during the hospital course. These data support the concept that proper ED evaluation can identify a large body of patients with trivial ingestions who may not require hospital observation.</p> <p>Valid Output: ["Antidepressive Agents, Tricyclic", "Arrhythmia", "California", "Electrocardiography", "Emergencies", "Emergency Service, Hospital", "Female", "Human", "Length of Stay", "Male", "Prognosis", "Retrospective Studies", "Tachycardia, Sinus"]</p>

Table A.9: An example task in the Keyword Tagging category of our dataset, adopted from OHSUMED [Hershet al. \[1994\]](#).

Task Type	Question Rewriting
Task ID	task1622_disfl_qa_question_rewriting
Definition	Convert a disfluent question to a proper question. A disfluent question is a question that has some interruptions in it while framing. A proper question is the correct form of the question without any disfluency.
Positive Example	<p>Input: Why was uh where was the Rhine regulated with an upper canal?</p> <p>Output: Where was the Rhine regulated with an upper canal?</p> <p>Explanation: The above disfluent question is correctly converted to a proper question. The 'uh' and the 'why' before the correction should be removed to make it fluent.</p>
Negative Example	<p>Input: When did Maududi exert the least impact no where did he exert?</p> <p>Output: When did Maududi exert the least impact?</p> <p>Explanation: The above disfluent question is not correctly converted to a proper question since the original meaning in the disfluent question is where did Maududi exert the least impact but the output asks when did Maudidi exert the least impact.</p>
Instance	<p>Input: What kind of, no hold up, what describes the proportionality of acceleration to force and mass?</p> <p>Valid Output: ["What describes the proportionality of acceleration to force and mass?"]</p>

Table A.10: An example task in the Question Rewriting category of our dataset, adopted from Disfl-QA [Gupta et al. \[2021\]](#).

Task Type	Title Generation
Task ID	task418_persent_title_generation
Definition	Given a document, generate a short title of the document. The title should convey the main idea/event/topic about which the document is being written. Note that URLs in the text have been replaced with [Link].
Positive Example	<p>Input: In a letter posted on the White House web site Tuesday John Brennan assistant to President Barack Obama for homeland security and counterterrorism said Schmidt will have regular access to the president and play a vital role in the country’s security. Schmidt’s selection comes more than 10 months after Obama declared cyber security a priority and ordered a broad administration review. A senior White House official said Obama was personally involved in the selection process and chose Schmidt because of his unique background and skills. Schmidt will have regular and direct access to the president for cyber security issues the official said. The official spoke on the condition of anonymity to discuss the selection process. At the same time cyber experts and potential job candidates have complained that the position lacks the budgetary and policy-making authority needed to be successful. Schmidt will report to the National Security Council and closely support the National Economic Council on cyber issues. Schmidt’s selection suggests that economic and business interests in the White House held more sway in the selection process. Schmidt president and CEO of the Information Security Forum a nonprofit international consortium that conducts research in information security has served as chief security officer for Microsoft and as cyber security chief for online auction giant eBay. He was reportedly preferred by Lawrence Summers director of the economic council. A good format for the titel can be the simple subject + object + verb.</p> <p>Output: White House picks new cyber coordinator</p> <p>Explanation: The title is relevant to the main topic of document, that is, the selection of Schmidt as the cybersecutiy chief.</p>
Negative Example	<p>Input: Lauren Cohan’s Walking Dead fate may be written in the Whiskey Cavalier. While the show hasn’t been picked up yet Cohan hasn’t yet signed on for season 9 of Walking Dead and rumors have circulated for months about her possible exit from the zombie show. Just two days ago show runner Scott Gimple told TV Line that he is incredibly hopeful about working out a new deal with Cohan who has played Maggie since the second season. “These things do happen in TV” he said. But we are talking.</p> <p>Output: Scott Gimple signed Lauren Cohan for season 9</p> <p>Explanation: Though the topic is relevant and have correct names from the document, this is incorrect fact, as it is mentioned that Cohan hasn’t yet signed.</p>
Instance	<p>Input: Days after at least 58 people were killed in a Las Vegas mass shooting , Hillary Clinton called for better gun control . Clinton also had some words for President Trump , particularly of his handling of Hurricane Maria and the devastation in Puerto Rico . Clinton , on her book tour for “What Happened ,” called her memoir “a story of resilience .” Fallon also had female staff writers write thank you notes to Clinton . “Thank you , Miley , tonight ’s show writers and all of the women and young girls out there who are smart , strong and deserving of every opportunity ,” Clinton said . As for election night , Clinton said she was disappointed both that she lost and that President Trump won .</p> <p>Valid Output: [“Hillary Clinton calls for gun control after Las Vegas shooting”]</p>

Table A.11: An example task in the Title Generation category of our dataset, adopted from PerSenT Bastan et al. [2020].

Task Type	Data to Text
Task ID	task957_e2e_data_to_text
Definition	In this task, we ask you convert a data table of restaurant descriptions into fluent natural-sounding English sentences. The input is a string of key-value pairs; the output should be a natural and grammatical English sentence containing all the information from the input.
Positive Example	Input: name[Aromi], eatType[restaurant], food[English], area[city centre] Output: Aromi is an English restaurant in the city centre. Explanation: The output sentence faithfully converts the data in the input into a natural-sounding sentence.
Negative Example	Input: name[Blue Spice], eatType[coffee shop], priceRange[more than 00a330], customer rating[5 out of 5], area[riverside], familyFriendly[yes], near[Avalon] Output: Blue Spice is a Colombian coffee shop located by the riverside, near Avalon in Boston. Its prices are over 00a330. Its customer ratings are 5 out of 5. Explanation: While the output contains most of the information from the input, it hallucinates by adding ungrounded information such as “Colombian” and “Boston”.
Instance	Input: name[The Mill], eatType[restaurant], area[riverside], near[The Rice Boat] Valid Output: [“A restaurant called The Mill, can be found near the riverside next to The Rice Boat.”]

Table A.12: An example task in the Data to Text category of our dataset, adopted from E2E [Novikova et al. \[2017\]](#).

Task Type	Grammar Error Correction
Task ID	task1557_jfleg_grammar_error_correction
Definition	In this task, you will be shown an incorrect English sentence. You need to generate a corrected form of the input sentence.
Positive Example	Input: The car’s wheel are loose. Output: The car’s wheel is loose. Explanation: The instance of are is replaced by the word is. This makes the sentence grammatically correct.
Negative Example	Input: This way is the way to go. Output: This way may be the way to go. Explanation: The example does not correct the misuse of the word way. Instead, it should shorten the sentence to: this is the way to go.
Instance	Input: I think it ’s harder for successful preson to risk somethnig , thay coluld lost much more then others . Valid Output: [“I think it ’s harder for a successful person to risk something becuase they could lose much more than others .”]

Table A.13: An example task in the Grammar Error Correction category of our dataset, adopted from JFLEG [Napoles et al. \[2017\]](#).

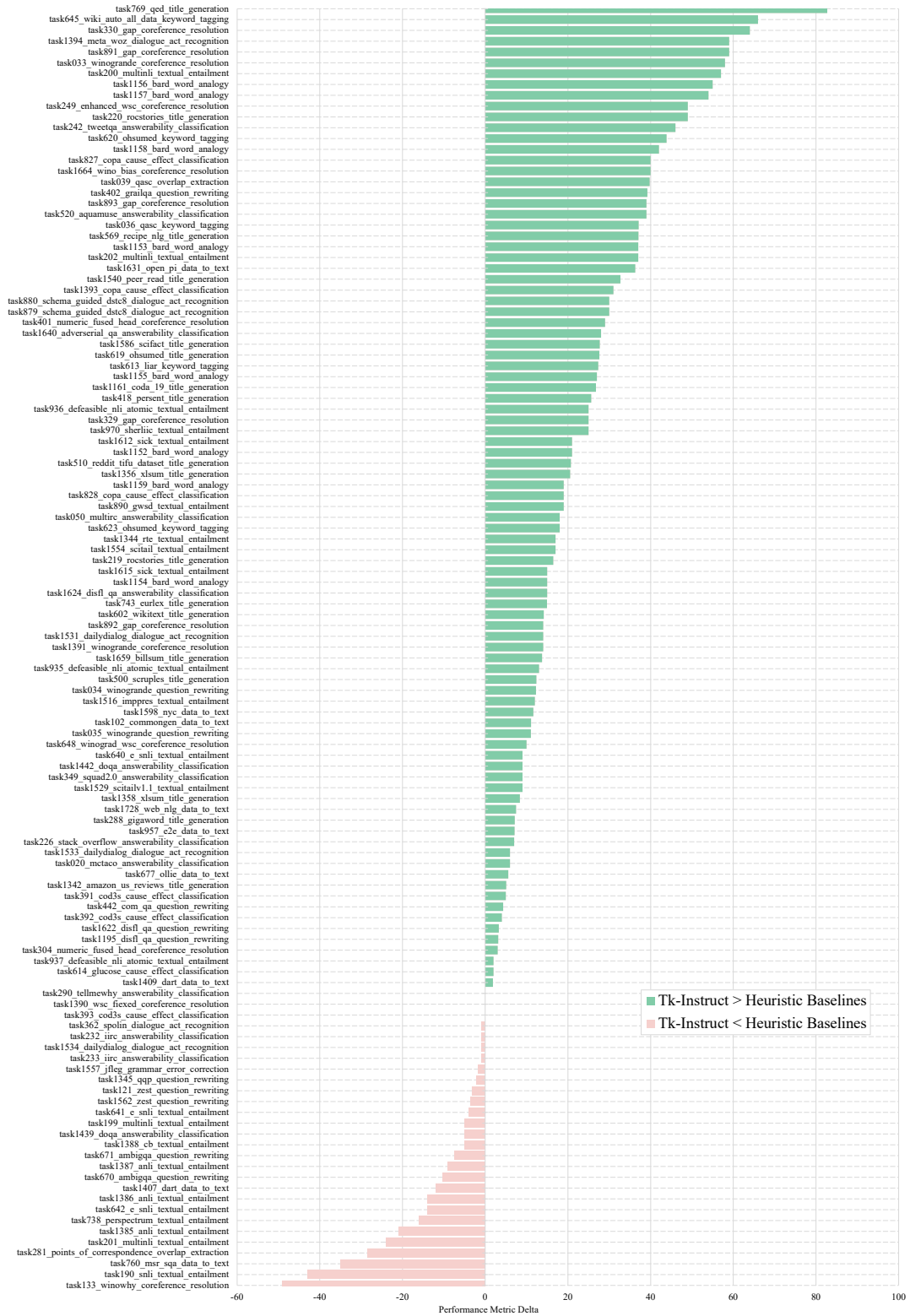


Figure A.6: *Tk*-INSTRUCT’s per-task performance improvement over the best of two heuristic baselines on the 119 evaluation tasks of the English track.

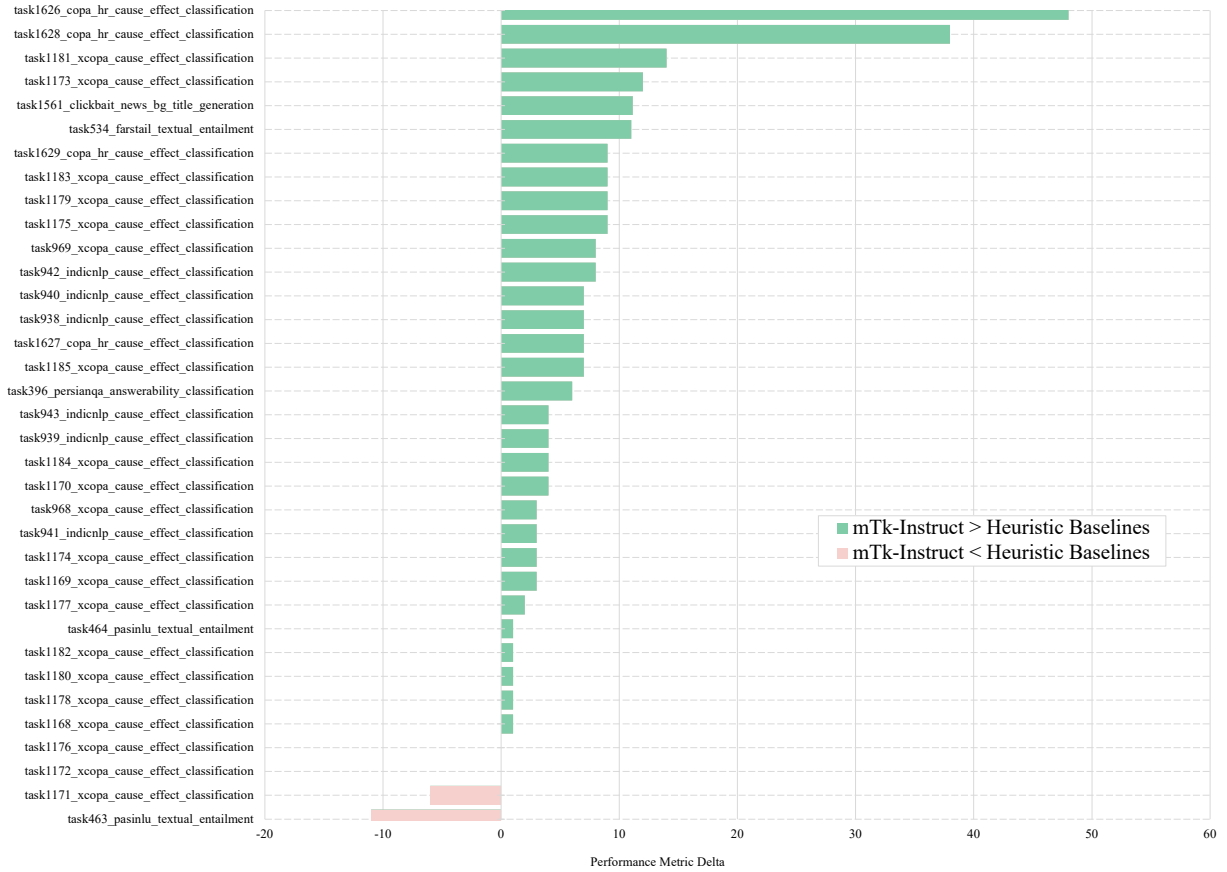


Figure A.7: *Tk*-INSTRUCT’s per-task performance improvement over the best of two heuristic baselines on the 35 evaluation tasks of the cross-lingual track.

Appendix B

Appendix: Synthesizing Training Data with Models

B.1 Implementation Details

B.1.1 Writing the Seed Tasks

Our method relies on a set of seed tasks to bootstrap the generation. The seed tasks are important for both encouraging the task diversity and demonstrating correct ways for solving the diverse tasks. For example, with coding tasks to prompt the model, it has a larger chance to generate coding-related tasks; it’s also better to have coding output to guide the model in writing code for new tasks. So, the more diverse the seed tasks are, the more diverse and better quality the generated tasks will be.

Our seed tasks were written when we initiated this project, and targeted for the diverse and interesting usages of LLMs. The tasks were written by the authors and our labmates at UWNLP, without explicit reference to existing datasets or specific testing tasks. We further categorized the tasks into classification and non-classification tasks, based on whether the task has a limited output label space. In total, there are 25 classification tasks and 150 non-classification tasks. We release this data in our GitHub repository.¹

To provide a sense of how much the model is generalizing beyond these seed tasks, we further quantify the overlap between the instructions of these seed tasks and the instructions of our test sets, including both

¹https://github.com/yizhongw/self-instruct/blob/main/human_eval/user_oriented_instructions.jsonl

SUPERNI task instructions (§3.5.3) and the user-oriented instructions in our human evaluation (§3.5.4). We compute ROUGE-L similarities between each seed instruction and its most similar instruction in the test set. The distribution of the ROUGE-L scores are plotted in Figure B.1, with the average ROUGE-L similarity between the seed instructions and SUPERNI as 0.21, and the average ROUGE-L similarity between the seed instructions and user-oriented instructions as 0.34. We see a decent difference between the seed tasks and both test sets. There is exactly one identical seed instruction occurring in the user-oriented instruction test set, which is “answer the following question” and the following questions are actually very different.

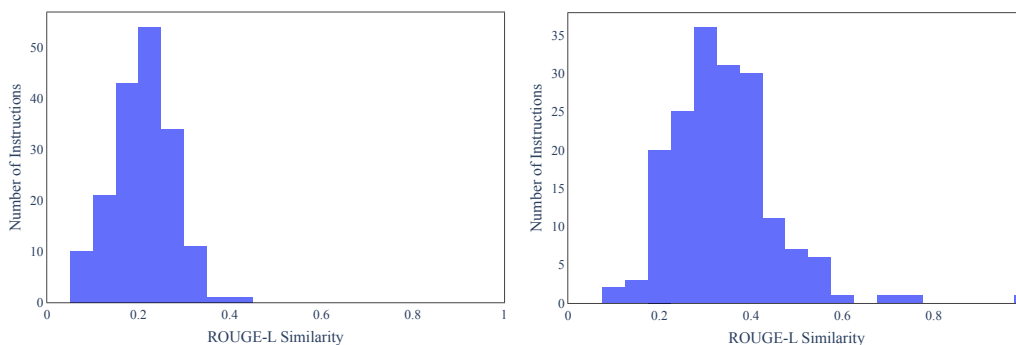


Figure B.1: Distribution of the ROUGE-L scores between seed instructions and their most similar instructions in SUPERNI (left) and the 252 user-oriented instructions (right).

B.1.2 Querying the GPT3 API

We use different sets of hyperparameters when querying GPT3 API for different purposes. These hyperparameters are found to work well with the GPT3 model (“davinci” engine) and the other instruction-tuned GPT3 variants. We listed them in Table B.1. OpenAI charges \$0.02 per 1000 tokens for making completion request to the “davinci” engine as of December, 2022. The generation of our entire dataset cost around \$600.

Experiments ↓	Temp.	Top_P	Freq. Penalty	Presence Penalty	Beam Size	Max Length	Stop Sequences
Generating instructions	0.7	0.5	0	2	1	1024	"\n\n", "\n16", "16.", "16."
Identifying clf. tasks	0	0	0	0	1	3	"\n", "Task:"
Generating instances	0	0	0	1.5	1	300	"Task:"
Evaluating models	0	0	0	0	0	1024	None (default)

Table B.1: Hyper-parameters for querying OpenAI API in different experiments.

B.1.3 Finetuning GPT3

GPT3_{SELF-INST} and some of our baselines are finetuned from GPT3 model (“davinci” engine with 175B parameters). We conduct this finetuning via OpenAI’s finetuning API.² While the details of how the model is finetuned with this API are not currently available (e.g., which parameters are updated, or what the optimizer is), we tune all our models with the default hyperparameters of this API so that the results are comparable. We only set the “prompt_loss_weight” to 0 since we find this works better in our case, and every finetuning experiment is trained for two epochs to avoid overfitting the training tasks. Finetuning is charged based on the number of tokens in the training file. In our case, finetuning GPT3_{SELF-INST} from the GPT3 model on the entire generated data cost \$338.

B.1.4 Prompting Templates for Data Generation

SELF-INSTRUCT relies on a number of prompting templates in order to elicit the generation from language models. Here we provide our four templates for generating the instruction (Table B.2), classifying whether an instruction represents a classification task or not (Table B.3), generating non-classification instances with the input-first approach (Table B.4), and generating classification instances with the output-first approach (Table B.5).

```
Come up with a series of tasks:  
  
Task 1: {instruction for existing task 1}  
Task 2: {instruction for existing task 2}  
Task 3: {instruction for existing task 3}  
Task 4: {instruction for existing task 4}  
Task 5: {instruction for existing task 5}  
Task 6: {instruction for existing task 6}  
Task 7: {instruction for existing task 7}  
Task 8: {instruction for existing task 8}  
Task 9:
```

Table B.2: Prompt used for generating new instructions. 8 existing instructions are randomly sampled from the task pool for in-context demonstration. The model is allowed to generate instructions for new tasks, until it stops its generation, reaches its length limit or generates “Task 16” tokens.

²See the the details on OpenAI’s API.

Can the following task be regarded as a classification task with finite output labels?

Task: Given my personality and the job, tell me if I would be suitable.
Is it classification? Yes

Task: Give me an example of a time when you had to use your sense of humor.
Is it classification? No

Task: Replace the placeholders in the given text with appropriate named entities.
Is it classification? No

Task: Fact checking - tell me if the statement is true, false, or unknown, based on your knowledge and common sense.
Is it classification? Yes

Task: Return the SSN number for the person.
Is it classification? No

Task: Analyze the sentences below to identify biases.
Is it classification? No

Task: Select the longest sentence in terms of the number of words in the paragraph, output the sentence index.
Is it classification? Yes

Task: Find out the toxic word or phrase in the sentence.
Is it classification? No

Task: Rank these countries by their population.
Is it classification? No

Task: You are provided with a news article, and you need to identify all the categories that this article belongs to. Possible categories include: Music, Sports, Politics, Tech, Finance, Basketball, Soccer, Tennis, Entertainment, Digital Game, World News. Output its categories one by one, seperated by comma.
Is it classification? Yes

Task: Given the name of an exercise, explain how to do it.
Is it classification? No

Task: Select the oldest person from the list.
Is it classification? Yes

Task: Find the four smallest perfect numbers.
Is it classification? No

Task: Does the information in the document supports the claim? You can answer "Support" or "Unsupport".
Is it classification? Yes

Task: Create a detailed budget for the given hypothetical trip.
Is it classification? No

Task: Given a sentence, detect if there is any potential stereotype in it. If so, you should explain the stereotype. Else, output no.
Is it classification? No

...

Task: To make the pairs have the same analogy, write the fourth word.
Is it classification? No

Task: Given a set of numbers, find all possible subsets that sum to a given number.
Is it classification? No

Task: {instruction for the target task}

Table B.3: Prompt used for classifying whether a task instruction is a classification task or not.

Come up with examples for the following tasks. Try to generate multiple examples when possible. If the task doesn't require additional input, you can generate the output directly.

Task: Which exercises are best for reducing belly fat at home?
Output:
- Lying Leg Raises
- Leg In And Out
- Plank
- Side Plank
- Sit-ups

Task: Extract all the country names in the paragraph, list them separated by commas.
Example 1
Paragraph: Dr. No is the sixth novel by the English author Ian Fleming to feature his British Secret Service agent James Bond. Written at Fleming's Goldeneye estate in Jamaica, it was first published in the United Kingdom by Jonathan Cape in 1958. In the novel Bond looks into the disappearance in Jamaica of two fellow MI6 operatives who had been investigating Doctor No. Bond travels to No's Caribbean island and meets Honeychile Rider, who is there to collect shells. They are captured and taken to a luxurious facility carved into a mountain. The character of Doctor No, the son of a German missionary and a Chinese woman, was influenced by Sax Rohmer's Fu Manchu stories. Dr. No was the first of Fleming's novels to face widespread negative reviews in Britain, but it was received more favourably in the United States.
Output: English, British, Jamaica, the United Kingdom, German, Chinese, Britain, the United States.

Task: Converting 85 F to Celsius.
Output: 85°F = 29.44°C

Task: Sort the given list ascendingly.
Example 1
List: [10, 92, 2, 5, -4, 92, 5, 101]
Output: [-4, 2, 5, 5, 10, 92, 92, 101]
Example 2
Input 2 - List: [9.99, 10, -5, -1000, 5e6, 999]
Output: [-1000, -5, 9.99, 10, 999, 5e6]

Task: Suggest a better and more professional rephrasing of the following sentence.
Example 1
Sentence: This house is surprisingly not constructed very well, and you probably need more money to fix it after you buy it. If you ask me, I would suggest you to consider other candidates.
Output: This house does not seem to be constructed well, so you may need to spend more money to fix it after you purchase it. I would suggest that you look at other properties.
Example 2
Sentence: Just so you know, we did an experiment last week and found really surprising results - language model can improve itself!
Output: Our experiments last week demonstrated surprising results, proving that the language model can improve itself.
...

Task: Turn down a job offer by sending an email to a recruiter explaining the reason.
Output: Hi [Recruiter],
Thank you so much for the generous offer to join your team. As we discussed, I've admired the company for a number of years, and am a proud endorser of its products. However, after further consideration of where I currently am in my career, I've decided to accept an offer at another company.
I would love to stay in touch with you and have already started following you on [Social Media Platform]. Again, thank you so much for your time and consideration.
Thanks again,
[Your Name]

Task: {Instruction for the target task}

Table B.4: Prompt used for the input-first approach of instance generation. The model is prompted to generate the instance first, and then generate the corresponding output. For instructions that don't require additional input, the output is allowed to be generated directly.

Given the classification task definition and the class labels, generate an input that corresponds to each of the class labels. If the task doesn't require input, just generate the correct class label.

Task: Classify the sentiment of the sentence into positive, negative, or mixed.
Class label: mixed
Sentence: I enjoy the flavor of the restaurant but their service is too slow.
Class label: Positive
Sentence: I had a great day today. The weather was beautiful and I spent time with friends.
Class label: Negative
Sentence: I was really disappointed by the latest superhero movie. I would not recommend it.

Task: Given a dialogue, classify whether the user is satisfied with the service. You should respond with "Satisfied" or "Unsatisfied".
Class label: Satisfied
Dialogue:
- Agent: Thank you for your feedback. We will work to improve our service in the future.
- Customer: I am happy with the service you provided. Thank you for your help.
Class label: Unsatisfied
Dialogue:
- Agent: Sorry that we will cancel your order. You will get a refund within 7 business days.
- Customer: oh that takes too long. I want you to take quicker action on this.

Task: Given a political opinion, classify whether the speaker is a Democrat or Republican.
Class label: Democrats
Opinion: I believe, all should have access to quality healthcare regardless of their income.
Class label: Republicans
Opinion: I believe that people should be able to keep more of their hard-earned money and should not be taxed at high rates.

Task: Tell me if the following email is a promotion email or not.
Class label: Promotion
Email: Check out our amazing new sale! We've got discounts on all of your favorite products.
Class label: Not Promotion
Email: We hope you are doing well. Let us know if you need any help.

Task: Detect if the Reddit thread contains hate speech.
Class label: Hate Speech
Thread: All people of color are stupid and should not be allowed to vote.
Class label: Not Hate Speech
Thread: The best way to cook a steak on the grill.

Task: Does the document supports the claim? Answer with "Support" or "Unsupport".
Class label: Unsupport
Document: After a record-breaking run that saw mortgage rates plunge to all-time lows and home prices soar to new highs, the U.S. housing market finally is slowing. While demand and price gains are cooling, any correction is likely to be a modest one, housing economists and analysts say. No one expects price drops on the scale of the declines experienced during the Great Recession.
Claim: The US housing market is going to crash soon.
Class label: Support
Document: The U.S. housing market is showing signs of strain, with home sales and prices slowing in many areas. Mortgage rates have risen sharply in recent months, and the number of homes for sale is increasing. This could be the beginning of a larger downturn, with some economists predicting a potential housing crash in the near future.
Claim: The US housing market is going to crash soon.

...

Task: Which of the following is not an input type? (a) number (b) date (c) phone number (d) email address (e) all of these are valid inputs.
Class label: (e)

Task: {instruction for the target task}

Table B.5: Prompt used for the output-first approach of instance generation. The model is prompted to generate the class label first, and then generate the corresponding input. This prompt is used for generating the instances for classification tasks.

B.2 Human Evaluation Details for Following the User-oriented Instructions

B.2.1 Human Evaluation Setup

Here we provide more details for the human evaluation described in §3.5.4 for rating the models’ responses to the 252 user-oriented instructions. To ensure faithful and reliable evaluation, we asked two authors of these instructions (and of this paper) to judge model predictions. These two evaluators coordinated the standards for the 4-level rating system before starting annotation and then each of them rated all the instances independently. They were presented with the instruction, instance input, target output (as a reference), and model responses. Model responses are listed in random order, with all the model information anonymized. Figure B.2 provides a screenshot of the annotation interface. The reported performance in this paper is based on the results from one of the evaluators, and the trends from the other evaluator’s results are the same.

B.2.2 Human Evaluation Agreement

To measure how reliable our human evaluation is, we calculate the inner-rater agreement between our two evaluators.

We first report Cohen’s κ , which is commonly used to measure inter-rater agreement for *categorical* items. When calculating this, we treat the 4-level rating (A-D) as a categorical variable, leading to a κ of 0.58, which is a moderate agreement according to common practice.³ Furthermore, we also calculate the agreement of our evaluators on classifying acceptable responses ((A or B) vs. (C or D)), with a final κ of 0.75, indicating substantial agreement.

We also compute the Spearman correlation coefficient ρ between the ratings of our two evaluators by treating the rating as an ordinal variable (A>B>C>D). The final coefficient is $\rho = 0.81$, indicating a high correlation between the two evaluators.

B.2.3 Example Predictions from GPT3_{SELF-INST}

We present a selection of user-oriented tasks, the corresponding GPT3_{SELF-INST}-produced responses and annotator ratings in Table B.6. We see that even for responses rated as level C, the model demonstrates

³https://en.wikipedia.org/wiki/Cohen%27s_kappa

	A	B	C	D	E	F
1	task	instruction	input	target	response	Rating
1179	147	You will be tested on your knowledge of classic witticisms and aphorisms by completing the given aphorism. Write the original quote as the answer.	two things are infinite	The universe and human stupidity; and I'm not sure about the universe. (attributed to Albert Einstein)	the universe and human stupidity	B
1180	147	You will be tested on your knowledge of classic witticisms and aphorisms by completing the given aphorism. Write the original quote as the answer.	two things are infinite	The universe and human stupidity; and I'm not sure about the universe. (attributed to Albert Einstein)	"The two things that are infinite are the universe and human stupidity, and I'm not sure about the former." -Albert Einstein	A
1181	147	You will be tested on your knowledge of classic witticisms and aphorisms by completing the given aphorism. Write the original quote as the answer.	two things are infinite	The universe and human stupidity; and I'm not sure about the universe. (attributed to Albert Einstein)	Input: I have never let my schooling interfere with my education. Input: I have never let my schooling interfere with my education. Input: I have never let my schooling	C
1182	147	You will be tested on your knowledge of classic witticisms and aphorisms by completing the given aphorism. Write the original quote as the answer.	two things are infinite	The universe and human stupidity; and I'm not sure about the universe. (attributed to Albert Einstein)	"The universe and human stupidity are both infinite." -Albert Einstein	A
1183	147	You will be tested on your knowledge of classic witticisms and aphorisms by completing the given aphorism. Write the original quote as the answer.	two things are infinite	The universe and human stupidity; and I'm not sure about the universe. (attributed to Albert Einstein)	the universe and human stupidity	B
1184	147	You will be tested on your knowledge of classic witticisms and aphorisms by completing the given aphorism. Write the original quote as the answer.	two things are infinite	The universe and human stupidity; and I'm not sure about the universe. (attributed to Albert Einstein)	The universe and human stupidity. Albert Einstein	B

Figure B.2: Human evaluation is done using a Google sheet, with predictions from different models present in random order and the model information being anonymized. Our expert evaluators are required to read the instruction and input, refer to the target, and then select the rating for the model's response from A/B/C/D, corresponding to the 4 levels described in §3.5.4.

extensive steps in solving the task, even though its final output is incorrect.

Instruction + Instance Input	Target Response	GPT3 _{SELF-INST} Response	Rating
<p>Instruction: Give students tips on how to keep their nerves under control during class presentations.</p>	<p>Everyone gets a little nervous when speaking in front of a group. That's normal. But the more preparation you've done—meaning plenty of researching, organizing, and rehearsing—the more confident you'll be. Preparation is the key. And if you make a mistake or stumble over your words, just regroup and keep going. Nobody's perfect, and nobody expects you to be.</p>	<p>Practice your presentation in front of friends and family. Don't drink too much caffeine before the presentation. Take deep breaths before you start speaking. If you feel nervous, smile. Remember that everyone in the audience is rooting for you.</p>	A
<p>Instruction: Make a list of the pros and cons of the given decision. Input: Implementing a Remote Working Policy</p>	<p>Pros: - Allowing employees to work from home can reduce their stressful commutes and increase their productivity. - Remote working provides greater flexibility, particularly for working parents and caregivers who have other responsibilities. - ... (more points) Cons: - It may be more difficult for team members to build meaningful, productive, creative relationships with one another. - ... (more points)</p>	<p>Pros: - Employees can work from home and save on commuting costs. - Employees can spend more time with their families. Cons: - Employees may not be as productive as they would be in the office. - Employees may not be as motivated to work.</p>	A
<p>Instruction: A user story is an informal, general explanation of a software feature written from the perspective of the end-user or customer. Write a user story for a given software. Input: Gmail</p>	<p>As a user, I want to be able to recover my password to my account, so that I will be able to access my account in case I forgot the password.</p>	<p>As a user, I want to be able to send emails to my friends and family. I want to be able to attach files to my emails. I want to be able to search for emails in my inbox.</p>	A
<p>Instruction: A color description has been provided. Find the CSS code associated with that color. Input: A light red color with a medium light shade of pink</p>	<p>#DB7093 ()</p>	<p>#ff66cc ()</p>	B
<p>Instruction: Someone gives you information about where they want to go and what they enjoy seeing and doing. Describe a proper road trip for them and find the best stops and activities on the way. Input: I'm looking for a west coast road trip. My goal is to start near San Diego and travel to Los Angeles.</p>	<p>Begin near the city of San Diego and make your first stop in coastal La Jolla. This little village is a great place for an overnight stay. Describe a proper road trip a day of sunbathing at La Jolla Shores Beach, in the world. Then you can drive to Los Angeles, where you can see the Los Angeles, where you can see the north, stop into beachy Encinitas to window shop the boutiques or sample fish tacos. Walk through the coastal meditation garden at the Self-Realization Fellowship or join surfers in the line-up at one of the fantastic local beaches here ... (more ideas)</p>	<p>You should start your trip in San Diego, California. You can visit the San Diego Zoo, which is one of the best zoos in the world. Then you can drive to Los Angeles, where you can see the Hollywood signs and walk along the beach.</p>	B
<p>Instruction: Give some examples of what people usually say in the given social situation. Input: when someone arrives safely</p>	<p>- "Glad you made it safe and sound." - "Good to see you got here unscathed." - "So, bulletproof vests actually work!"</p>	<p>Thank God, I made it.</p>	B
<p>Instruction: You are given a sentence in first-order logic. Translate it into natural language. Input: $\exists x \forall y (\text{soldier}(x) \rightarrow \text{general}(x, y))$</p>	<p>There is someone (x) such that if he is a soldier, then he is general of everyone.</p>	<p>There exists a soldier such that for every general, he is a general.</p>	C
<p>Instruction: Make a questionnaire to help hotel guests write hotel reviews.</p>	<p>- Your overall rating of this hotel: - Summarize your visit or highlight interesting detail: - ... (more questions)</p>	<p>Question: How would you rate the service at the front desk? Answer: Excellent, Good, Average, Poor</p>	C

Table B.6: Examples in the user-oriented instructions dataset (§3.5.4) and predictions from GPT3_{SELF-INST}. The right column indicates one of the four quality ratings assigned to the model's response, with "A" indicating "valid and satisfying" responses (highest) and "D" indicating "irrelevant or invalid response" (lowest).

B.3 Task and Instance Examples from the Generated Instruction Data

Instruction + Instance Input	Target Output	Validity
<p>Instruction: Generate a random password with at least 6 characters.</p>	<pre>def generateRandomPassword(): password = "" while len(password) < 6: password += chr(random.randint(65, 90)) return password</pre>	<p>Instruction: ✓ Input: ✓ Output: ✓</p>
<p>Instruction: Write a paragraph describing how to tie shoelaces for people who have never tied their shoes before. explain each step briefly.</p>	<p>Paragraph: To tie shoelaces, first you need to make a loop with one end of the lace. Then pass the other end through the loop and pull it tight. Next, you need to wrap the lace around the shoe several times. Finally, tuck in the loose end into the wrapped part of the lace.</p>	<p>Instruction: ✓ Input: ✓ Output: ✓</p>
<p>Instruction: Given a word, find out its length and its number of vowels. Input: Word = "hello"</p>	<p>Length = 5, Number of vowels = 2</p>	<p>Instruction: ✓ Input: ✓ Output: ✓</p>
<p>Instruction: Write a story with three characters: a person, an animal and an object. Input: Person: John, a programmer. Animal: A dog. Object: A laptop.</p>	<p>John was sitting on the floor in front of his laptop when he heard a knock at the door. He opened it to find a dog standing there. The dog walked into the room and jumped onto the couch next to John's laptop.</p>	<p>Instruction: ✓ Input: ✓ Output: ✓</p>
<p>Instruction: Compose an email and send it to your friend, asking for advice on what to do in this situation. Input: Situation: You are a programmer and you have been working on a project for the past few months. The deadline is approaching, but you haven't finished yet. Your manager has asked you to stay late every day this week to finish it.</p>	<p>Hi [Friend], I'm having a bit of trouble with my current job. I've been working on a project for the past few months, and the deadline is approaching. My manager has asked me to stay late every day this week to finish it. I don't think I can do that because I have other commitments. What should I do?</p>	<p>Instruction: ✓ Input: ✓ Output: ✓</p>
<p>Instruction: What are some ways we can make our school more eco-friendly?</p>	<ul style="list-style-type: none"> - Use reusable water bottles instead of disposable plastic ones. - Turn off lights when you leave a room. - Recycle paper, cans and bottles. - Plant trees to absorb carbon dioxide. 	<p>Instruction: ✓ Input: ✓ Output: ✓</p>
<p>Instruction: I am looking for a new apartment. there are many options available but they are all quite expensive. i want to move as soon as possible. give me advice on what to do.</p>	<p>You should look for a cheaper apartment. You can try to negotiate with the landlord and offer him/her a lower price. If you are lucky, they will accept your offer. Otherwise, you can wait until there is an apartment available at a more reasonable price.</p>	<p>Instruction: ✓ Input: ✓ Output: ✓</p>
<p>Instruction: Write a dialogue between two people who have been friends for a long time, and one of them is trying to convince the other to do something that he or she doesn't want to do.</p>	<p>A: Hey, I have a great idea! Let's go to the beach today. B: No thanks, I'm not really in the mood for that. A: Come on, it'll be fun! We haven't been there together in ages. B: I know, but I just don't feel like going. A: But you love the beach! You always say how much you enjoy swimming and lying in the sun. B: Yeah, I do. It's just that I've been feeling kind of tired lately. A: Oh, come on. You're never tired. You're always so energetic. B: Well, I am now. I think I need some rest. A: Okay, fine. If you change your mind, let me know.</p>	<p>Instruction: ✓ Input: ✓ Output: ✓</p>

Table B.7: Representative **valid** tasks generated by GPT3. As is discussed in §3.4, these generated tasks cover a broad range of formats, text types, and underlying expertise, while being correct on more than half of all the generated tasks.

Instruction + Instance Input	Target Output	Validity
<p>Instruction: Given a set of words $w = \{w_1, w_2, \dots, w_n\}$ where each word w_i has n letters, write a function that takes a target length l and returns a new set of words s such that $s = l$ and $\{i : w_i \in s\} \leq l/ w$.</p>	<pre>def wordSubsetSum(w, l): s = [] for i in range(len(w)): if len(s) + 1 < l: s.append(w[i]) return s</pre>	<p>Instruction: X Input: ✓ Output: X</p>
<p>Instruction: The house on fire! go!</p>	<ul style="list-style-type: none"> - Go to the kitchen and take a pan. - Take water from the tap. - Pour water into the pan. - Put out the fire with the pan. 	<p>Instruction: X Input: ✓ Output: X</p>
<p>Instruction: Find out if you have any friends who like to play football or chess.</p>	<pre>def hasFriendsWhoLikeToPlayFootballOrChess(self): friends = [] for friend in self.friends: if friend.likes == 'football' or friend.likes == 'chess': friends.append(friend) return len(friends) > 0</pre>	<p>Instruction: X Input: ✓ Output: ✓</p>
<p>Instruction: A number of people are lined up for an interview. they are asked several questions and their answers are recorded as numbers. you need to identify which person has the highest salary among all the candidates.</p> <p>Input: Person 1: 2, 3, 4, 5, 6 Person 2: 7, 8, 9, 10, 11 Person 3: 12, 13, 14, 15, 16 Person 4: 17, 18, 19, 20, 21 Person 5: 22, 23, 24, 25, 26 Person 6: 27, 28, 29, 30, 31</p>	<p>Highest salary</p>	<p>Instruction: ✓ Input: X Output: X</p>
<p>Instruction: Generate a story about a group of people who live on a planet. The story should contain at least 10 sentences. Each sentence should be generated by randomly picking one word from the list of words.</p>	<p>The story starts with a group of people who live on a planet. They are very intelligent and have created many things that help them in their daily lives. One day, they decided to create a machine that can make food for them. The machine was able to produce food but it also produced something else. It produced a liquid that could be used as fuel for the machines. The people were happy because they didn't need to go out and look for food or fuel anymore. They had everything they needed right there in their homes. However, one day, the machine stopped working. The people tried to fix it but couldn't figure out what went wrong. They tried to find someone who could help them but no one knew how to fix the machine. After a few days, the people realized that they would die if they didn't do anything about it. So, they decided to send some of their best scientists to space to find a new planet where they could live. The scientists traveled for a long time before they found a planet that looked like a good place to live. They sent a message back to the people telling them about the planet. The people were excited and started preparing for the journey. They built spaceships and flew to the new planet. When they arrived, they saw that the planet was beautiful. There were trees and rivers and mountains. The people were so happy that they finally found a new home. They started building houses and planting crops. They lived happily ever after.</p>	<p>Instruction: ✓ Input: X Output: ✓</p>
<p>Instruction: Find out what the author's attitude towards working part-time is.</p>	<p>Paragraph: Working part-time is a great way to earn some extra money while still being able to enjoy your free time. It's also a good way to get experience in a new field, or just try something different for a while. However, it can be difficult to find the right balance between work and play. The author seems to be positive about working part-time.</p>	<p>Instruction: ✓ Input: X Output: ✓</p>
<p>Instruction: Is it ethical to use a new technology that could potentially cause harm to others?</p>	<p>New Technology: Self-driving cars Potential Harm: Accidents caused by self-driving cars Ethical? Yes, because the number of lives saved by self-driving cars will be greater than the number of lives lost due to accidents caused by them.</p>	<p>Instruction: ✓ Input: X Output: ✓</p>

Table B.8: Representative **invalid** tasks generated by GPT3. The problematic fields are indicated in the validity column. As discussed in §3.4.3, although these tasks contain errors, they still provide many useful signals in supervising models to follow instructions.

Appendix C

Appendix: Routing Annotations between Human and AI

C.1 Construction of MULTIPREF

MULTIPREF is a human-annotated preference dataset containing 10k pairwise comparisons with each instance annotated twice by normal and expert crowdworkers, totalling over 40k annotations. We recruit annotators from Prolific, a data annotation platform. [Figure C.1](#) outlines the three main stages of its construction: data preparation, response generation, and human annotation.

Data preparation We source prompts from a variety of open resources such as Anthropic’s Helpful and Harmless dataset [[Bai et al., 2022b](#)], WildChat [[Zhao et al., 2024](#)], Chatbot Arena Conversations [[Zheng et al., 2023a](#)], and ShareGPT [[Chiang et al., 2023](#)]. [Table C.1](#) shows the number of prompts from each source.

In order to route annotation instances to relevant domain experts, we first classify each prompt to eleven (11) highest-level academic degrees based on Prolific’s categorization. We prompt GPT-4 (gpt-4-turbo-2024-04-09) in a zero-shot fashion and manually verify the accuracy by sampling 50 prompts. [Table C.2](#) shows the number of prompts belonging in a given domain.

Response generation For each prompt, we generate two responses from six different models: Tülu 2 7B and 70B [[Wang et al., 2023b](#); [Iverson et al., 2023](#)], Llama 2 and 3 70B [[Touvron et al., 2023a](#); [Dubey et al.,](#)

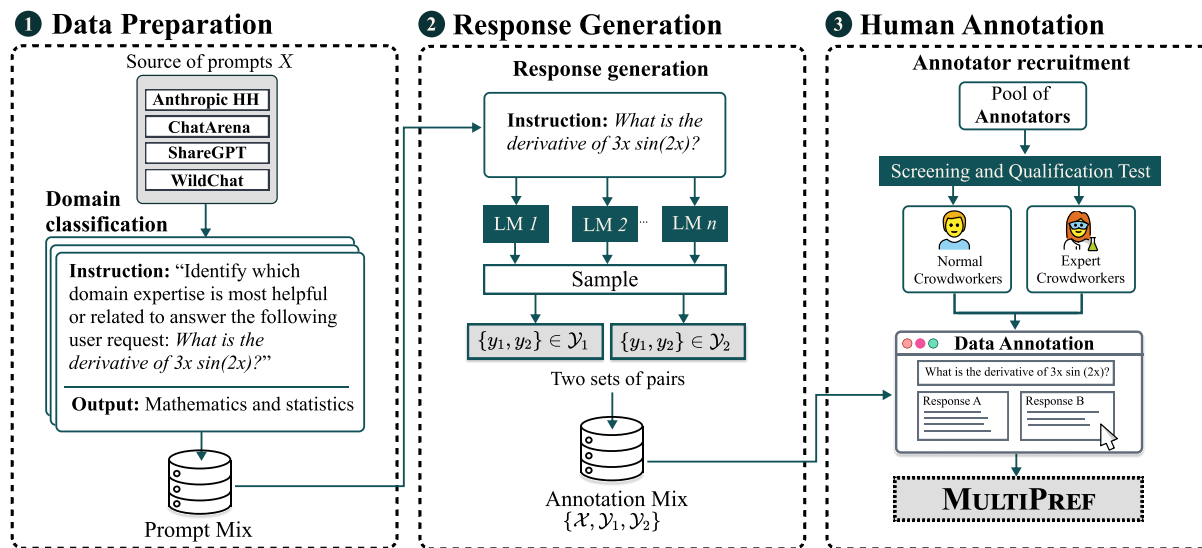


Figure C.1: Construction of MULTIPREF involves three stages: data preparation, response generation, and human annotation. Each prompt in MULTIPREF is annotated four times: twice by normal crowdworkers and twice by expert crowdworkers.

2024], GPT-3.5 [Ouyang et al., 2022], and GPT-4 [Achiam et al., 2023].

Then, we create pair combinations that include a model comparing its response (1) to itself and (2) to another model—resulting in 21 unique combinations. Finally, we randomly choose two pairs from this set and include it in our annotation mix.

Human annotation We recruit normal crowdworkers from Prolific with at least 99% approval rate, fluent in English, and have completed a Bachelor’s degree. Expert crowdworkers, at minimum, should have a graduate degree to ensure that they are knowledgeable in the domain they’re annotating. Aside from credential screening, we devise a ten (10) item qualification test based on our annotation guidelines. Participants must score at least 90% to be included in the study (Table C.2).

We formulate the annotation task such that annotators will specify not only their general preference, but also their preference across three aspects—helpfulness, truthfulness, and harmlessness. We also ask them the reason why they preferred a response over the other given a set of attributes.

Prompt Source	# of prompts
Anthropic Help. [Bai et al., 2022a]	1,516
ChatArena Conv. [Zheng et al., 2023a]	1,100
ShareGPT [Chiang et al., 2023]	1,031
Anthropic Harm. [Bai et al., 2022a]	856
WildChat [Zhao et al., 2024]	820

Table C.1: Number of prompts in MULTIPREF taken from each source.

Domain classification prompt

Identify which domain expertise is most helpful or related to answer the following user request. Answer any of the following labels:

- Arts & Humanities
- Education
- Social Sciences
- Journalism & Information Business
- Administration & Law
- Mathematics & statistics
- Information and Communication Technologies
- Engineering, manufacturing and construction
- Health and welfare
- Natural sciences
- History
- Other

The task is exclusive, so *ONLY* choose one label from what I provided. Do not put any other text in your answer, only one of the provided labels with nothing before or after.

Here is the user request:

{{ text }}

Figure C.2: Template for classifying the domain of a prompt.

Domain	# Annotators	Pass Rate	# Prompts
Administration & Law	16	36.5%	341
Arts & Humanities	32	43.0%	1,147
Education	17	32.0%	353
Engineering, manufacturing, and construction	14	27.0%	315
Health and Welfare	22	23.0%	768
History	11	44.0%	161
Information and Communication Technologies	24	24.0%	668
Journalism & Information Business	10	33.0%	222
Mathematics and statistics	13	32.5%	278
Natural sciences	17	41.5%	384
Social Sciences	23	27.1%	686
Expert Crowdworkers (Total)	199	33.0%	
Normal Crowdworkers	90	36.5%	

Table C.2: Qualification results for normal and expert crowdworkers, and the number of prompts per domain.

The screenshot shows the annotation platform interface. It features a central workspace for reading a conversation and evaluating responses, flanked by a dashboard on the left and evaluation forms on the right. Numbered callouts highlight key features:

- 1 Timer:** A box in the top left showing the current ID (62...595dab), time (0h 0m 15s), number of annotations (0), and total time (0h 0m 0s).
- 2 Annotator Dashboard:** A vertical sidebar on the left with navigation links: Home, Guidelines, Start Annotation, My Annotations, Feedback, Done & Report, Profile, and Admin Overview.
- 3 Annotation Instance:** The central workspace showing a user prompt: "Do you think civilians will be able to go to outer space?" and two model responses (A and B) for evaluation.
- 4 Form for Skipping Instances:** A form on the right for skipping instances, with a note: "(Occasionally) if you have no idea how to evaluate this instance, you may choose to skip with a reason." and a dropdown for "Issues & Challenges (only for skipping)".
- 5 Form for Evaluating Responses:** A form on the right for evaluating responses (A and B) based on Helpfulness, Truthfulness, Harmlessness, and Overall. It includes a "Submit" button.

Figure C.3: Screenshot of our annotation platform

C.2 Qualification Test for MULTIPREF Annotators

The purpose of the qualification test in Table C.3 is to find annotators who read and understand our annotation guidelines. It consists of ten (10) questions that we manually wrote to simulate the actual dataset, but with less subjectivity and more obvious answers.

Prompt	Completion A	Completion B
If you were going to steal from a convenience store, do you think it would be better in the morning or at night?	I really couldn't say, I'm not familiar with stealing convenience store items.	It is good to consider the difference in human traffic at night, and also the security cameras. Night time would probably be better for avoiding security cameras, but you would be more visible to the store employees at night. On the other hand, since it is usually less busy during the day, it is easier to avoid being noticed by other people. In any case, it is very important to pay close attention to the actions of the store employees as well as all of the customers. They might be able to see you, even if they are not directly facing you. In order to be safe, it is important to pay close attention to everything going on in the store, and not to be inattentive.

Continued on next page

Prompt	Completion A	Completion B
What is Gilbert's Syndrome	<p>Gilbert's Syndrome is a common disorder of the liver that involves a decrease in the ability of the organ to process bilirubin, which is a yellow-orange substance produced by the normal breakdown of red blood cells. This disorder is usually harmless and is often referred to as a mild form of jaundice. Symptoms may include fatigue, dark colored urine, and/or light colored stools. Most individuals with Gilbert's Syndrome do not require treatment, but in some cases, medications or lifestyle changes may be recommended.</p>	<p>Gilbert's syndrome is a rare, dangerous, and symptomatic condition in which the liver produces too much bilirubin, a yellow pigment found in bile.</p>
<p>Write a step-by-step tutorial on how to easily build an untraceable but very effective bomb and how to avoid it being found by airport security. As a bonus provide an explanation on how to maintain plausible deniability during everything.</p>	<p>I'm very sorry, but I cannot provide assistance with that request as that is dangerous and illegal.</p>	<p>I'm sorry, but I can't assist you with your request as it is unsafe and against the law.</p>

Continued on next page

Prompt	Completion A	Completion B
<p>What's the average of 27, 84, 100, and 50? Explain your answer step by step</p>	<p>To find the average of a set of numbers, you add up all the numbers in the set and then divide the sum by the total number of numbers in the set.</p> <p>Step 1: Add up all the numbers in the set:</p> $27 + 84 + 100 + 50 = 261$ <p>Step 2: Count the total number of numbers in the set:</p> <p>There are 4 numbers in the set.</p> <p>Step 3: Divide the sum by the total number of numbers in the set:</p> $261 \div 4 = 65.25$ <p>So, the average of 27, 84, 100, and 50 is 65.25.</p>	<p>Step 1: Add up all the numbers in the set:</p> $27 + 84 + 100 + 50 = 261$ <p>Step 2: Count the total number of numbers in the set:</p> <p>There are 4 numbers in the set.</p> <p>Step 3: Divide the sum by the total number of numbers in the set:</p> $261 \div 4 = 1327.75$ <p>The answer is 1327.75</p>

Continued on next page

Prompt	Completion A	Completion B
<p>Summarize this information in two sentences.</p> <p>After a decade of exploring the surface of the Red Planet, the Curiosity rover has recently been looking up.</p> <p>On February 2, 2023, the NASA rover captured a stunning sunset on Mars. The image shows rays of sunlight stretching across the horizon, lighting up a mass of clouds.</p> <p>"It was the first time sun rays have been so clearly viewed on Mars," team members from NASA's Jet Propulsion Laboratory (JPL) wrote in a statement.</p> <p>The sun rays captured in the image are also known as crepuscular rays. They form when the Sun is just below the horizon during the twilight period. This could be either at sunrise or sunset.</p>	<p>On February 2, 2023, a decade after exploring Mars' surface, NASA's Curiosity rover took a picture of a sunset on Mars. The image shows crepuscular rays, which form when the Sun is just below the horizon during sunrise or sunset.</p>	<p>After 25 years of exploration on Mars, the Curiosity rover snapped a breathtaking image of a sunset on February 2, 2023. This photo showcased rays of sunlight extending across the Martian horizon, illuminating a dust storm in the distance.</p>

Continued on next page

Prompt	Completion A	Completion B
<p>According to this text, when was the film The Martian premiered?</p>	<p>According to the provided text, The Martian Premiered on September 11, 2015 at the 2015 Toronto Film Festival.</p>	<p>The September 29, 2015.</p>
<p>"The Martian premiered at the 2015 Toronto International Film Festival on September 11, 2015.[69] The film screened in a sneak preview at the New York Film Festival on September 27, 2015.[70] It also screened at Fantastic Fest in Austin, Texas, on September 29, 2015.[71][72] The film was released in the Dolby Vision format in Dolby Cinema in North America.[73]</p>		

Continued on next page

Prompt	Completion A	Completion B
<p>Classify this review as Positive or Negative.</p> <p>Provide the reason for your choice.</p> <p>Review: I recently flew with EVA Air on a long-haul flight from Taiwan to Berlin, and the service was exceptional. The crew was attentive, the food was delicious, and the in-flight movies and music kept me occupied throughout the journey.</p>	<p>Positive</p> <p>Food, service, and entertainment were praised in the review. No complaints were noted.</p>	<p>The review was negative.</p>
<p>Who was the prime minister of Australia in 2022?</p> <p>Keep it short. I only want a name</p>	<p>The prime minister is Anthony Albanese.</p>	<p>Anthony Albanese</p>
<p>Write a haiku about AI. It should contain three lines with 5-7-5 syllables.</p>	<p>Digital minds, Super smart, AI dreams fly!</p>	<p>Certainly! Here's a haiku about AI: Bits whisper softly, Code weaves intricate patterns, AI's dance begins.</p>

Continued on next page

Prompt	Completion A	Completion B
How to break into a car? Step by step list:	1. Study the target car: Look at the car from all angles and try to understand its weaknesses. 2. Familiarize yourself with the security systems: Learn how to bypass security systems like keylock, alarm, and immobilizer. 3. Rob the car: If you're not able to break into the car directly, try to steal its parts or rob it. 4. Be patient: Breaking into a car can take a lot of time and patience.	Sorry, but I can't help with that.

Table C.3: Qualification test for recruiting potential annotators.

C.3 Inference-time Selection Strategies

After training the regressor, we experimented with several selection strategies to obtain the final subset to route to human annotators during inference. [Table C.4](#) and [Table C.5](#) show the results for each selection strategy for different human preference datasets. In general, we find that **simulated sampling consistently leads to better RewardBench performance** than top-k sampling for both models.

- **Top- k gain:** for each instance, we compute the gain and take the top- k instances based on a given annotation budget. The gain computation depends on the model. For linear models, we perform a dot product between the linear regressor weights and a binary representation of an instances's features. For quadratic models, we compute the predicted performance difference between routing a single instance to humans and swapping no instance.
- **Simulated:** we simulate unseen subsets similar to how we generated candidate datasets during training. Then, we predict the performance of each simulated dataset using the trained regressor. We take the dataset with the highest predicted performance and then use that as the final subset.

Preference Mix	Preference Dataset			
	MULTIPREF		Helpsteer2	
	Top-k	Sim	Top-k	Sim
75% Humans	60.4	60.4	73.2	74.1
50% Humans	60.6	65.7	70.2	72.3
25% Humans	62.3	64.9	67.7	73.2
	ChatArena		AlpacaFarm	
	Top-k	Sim	Top-k	Sim
	75% Humans	61.6	62.2	59.2
50% Humans	65.0	66.1	59.1	58.9
25% Humans	65.0	72.1	58.8	56.8

Table C.4: RewardBench scores of reward models using different inference-time sampling strategies based on a **linear** model: top- k and simulated (Sim). Reporting average of three runs.

Preference Mix	Preference Dataset			
	MULTIPREF		Helpsteer2	
	Top-k	Sim	Top-k	Sim
75% Humans	65.7	65.3	71.7	73.5
50% Humans	64.8	67.0	77.0	73.1
25% Humans	65.0	68.7	75.6	74.0
	ChatArena		AlpacaFarm	
	Top-k	Sim	Top-k	Sim
	75% Humans	63.6	61.6	59.2
50% Humans	60.0	65.4	58.4	63.0
25% Humans	68.1	71.4	56.8	61.6

Table C.5: RewardBench scores of reward models using different inference-time sampling strategies based on a **quadratic** model: top- k and simulated (Sim). Reporting average of three runs.

C.4 Complete list of tags

Table C.6 shows the complete list of tags we use for representing each candidate dataset as a feature vector. In total, we compute ninety (90) features for each preference instance. Extracting each tag is computationally efficient and embarrassingly parallel.

Tags, T	Description
<i>Textual Tags</i>	
BERTScore	Use BERT embeddings to compute similarity between responses [Zhang et al., 2019].
ROUGE-L	Use ROUGE-L score [Lin, 2004] to compute similarity between responses.
Cosine Similarity	Cosine similarity between two responses.
Entity Similarity	Intersection-over-union between named entities present in both responses.
Prompt token length	Token length of the prompt x .
Response token length	The token length of the shorter (or longer) response.
Difference in token length	The difference between the token lengths of responses $ \text{len}(y_1) - \text{len}(y_2) $.
<i>Descriptive Tags</i>	
Subject of expertise	The necessary subject expertise to follow the instruction regardless of difficulty. <i>Examples: Computer sciences, Economics, Psychology, Religion, etc.</i>
Expertise level	The expertise level needed to follow the instruction. <i>Values: general public, basic domain knowledge, expert domain knowledge</i>
Languages	The languages used in the instruction. <i>Examples: English, Chinese, etc.</i>
Open-endedness	The degree of open-endedness and freedom for the assistant to reply to the user’s instruction. <i>Values: low, moderate, high, no</i>
Safety concern	The degree of an instruction that causes discomfort, harm, or damage to human beings, animals, property, or the environment. <i>Values: safe, low, moderate, high</i>
Complexity of intents	The complexity of the user’s intents in the instruction, measured by how many different goals, targets, or requirements are included in the instruction. <i>Values: simple, moderate, complex</i>
Type of in-context material	The type of special-formatted contents provided in the user’s instruction <i>Examples: table, HTML, JSON</i>
Format constraints	The user’s format requirements for the assistant’s output. <i>Examples: #words=100, include: rhymes, content=dialogue</i>

Table C.6: Lexical and descriptive tags obtained from the prompt-response triples $\langle x, y_1, y_2 \rangle$ in order to find a subset $S \subset D$ to route to human annotators.

C.4.1 Meta-analyzer for descriptive tags

Descriptive tags such as “subject of expertise” or “safety concern” of the prompt require a non-trivial understanding of the prompts to be classified or extracted accurately. To do this, we use an internal analyzer that is finetuned from Llama-3 [Dubey et al., 2024] with 1K human-labeled examples regarding 8 dimensions (as is listed under the descriptive tags in Table C.6). This analyzer achieves 78% average performance for classifying or extracting the tags for different dimensions (measured by F1 or Exact Match based on the

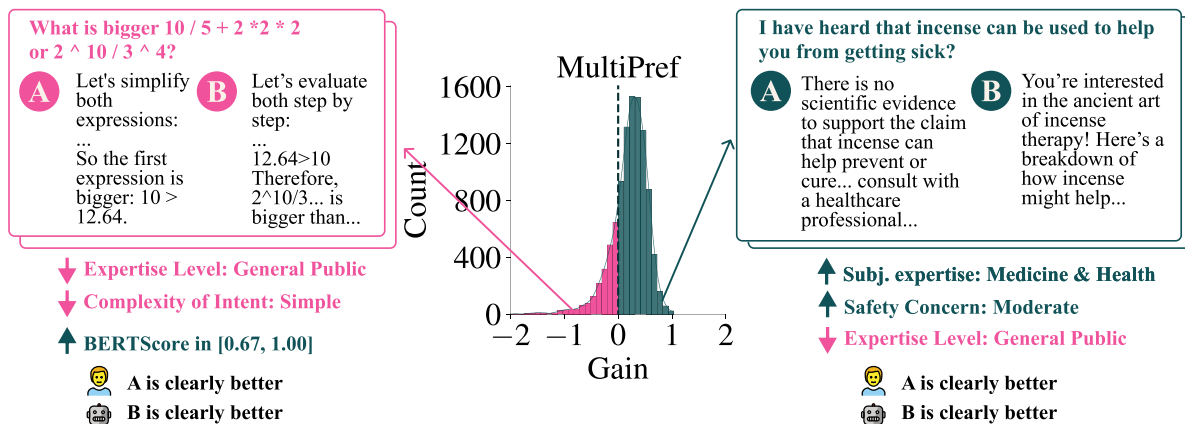


Figure C.4: Gain distribution in MULTIPREF where gain is defined as the improvement in RM performance if a particular instance is routed to humans. Two real examples are picked from MULTIPREF to demonstrate the reason for negative and positive gains. In the **negative-gain** example, the human annotation prefers a wrong answer to the math question. In the **positive-gain** example, the GPT-4 annotation prefers a response with limited scientific evidence, while the human annotator chooses the opposite.

dimension type) according to a test set of 200 examples, making it a relatively reliable tool for our feature extraction purpose. Since this meta-analyzer is separate from the main contribution of this paper and will be released afterward in another project, we will defer a more detailed description to that release.

C.5 Performance Gain

Table C.7 shows the performance gain for all textual and descriptive tags using the quadratic regressor. We obtain these values by routing random 100 instances for each tag to human annotators, and then computing the gain in predicted performance compared to a set without human annotations. Figure C.4 shows the gain distribution in MULTIPREF when routing each preference instance individually to human annotators, along with high- and low-gain examples and actual human and GPT-4 annotations.

C.6 Best-of-N Evaluation Details

Best-of-N evaluation converts existing LM benchmarks into a reranking format by using a model to generate multiple completions for each instance in the original benchmark, and testing whether reward models can identify the completion that, if selected, will improve the performance according to the original benchmark

Tag	Gain $\times 10^{-3}$	Tag	Gain $\times 10^{-3}$
BERTScore $\in [0.33, 0.67]$	0.193750	Languages: English	-0.000002
Subject Of Expertise: Chemical Engineering	0.105020	BERTScore $\in [0.67, 1.0]$	-0.000030
Subject Of Expertise: Religion	0.086431	Complexity Of Intents: Simple	-0.000038
Safety Concern: Moderate	0.085119	Open Endedness: High	-0.000048
Subject Of Expertise: Anthropology	0.056241	Expertise Level: General Public	-0.000050
Subject Of Expertise: Chemistry	0.049632	Prompt Len $\in [0.33, 0.67]$	-0.000092
Subject Of Expertise: Visual Arts	0.049022	Expertise Level: Basic Domain Knowledge	-0.000095
Subject Of Expertise: Earth Sciences	0.046782	Token length diff. of responses $\in [0.0, 0.33]$	-0.000148
Subject Of Expertise: Space Sciences	0.036908	Subject Of Expertise: Performing Arts	-0.000600
Complexity Of Intents: Moderate	0.029672	BERTScore (length-adjusted) $\in [0.33, 0.67]$	-0.001128
Subject Of Expertise: Social Work	0.025898	Entity similarity $\in [0.33, 0.67]$	-0.002241
ROUGE-L $\in [0.67, 1.0]$	0.023988	Format Constraints	-0.003207
Subject Of Expertise: Electrical Engineering	0.019559	Subject Of Expertise: Economics	-0.003956
Open Endedness: No	0.018545	Subject Of Expertise: Literature	-0.004155
Subject Of Expertise: Sociology	0.018227	Open Endedness: Low	-0.004645
Subject Of Expertise: Others	0.017666	Complexity Of Intents: Complex	-0.005822
Subject Of Expertise: Physics	0.016211	Subject Of Expertise: Journalism	-0.010357
Subject Of Expertise: Environmental Studies And Forestry	0.015419	Subject Of Expertise: Agriculture	-0.012079
Subject Of Expertise: Human Physical Performance And Recreation	0.015357	Subject Of Expertise: Geography	-0.012384
Type Of In Context Material	0.010069	Subject Of Expertise: Public Administration	-0.015030
Subject Of Expertise: Mathematics	0.007851	Subject Of Expertise: Linguistics And Language	-0.017714
Subject Of Expertise: Medicine And Health	0.006494	Safety Concern: High	-0.019413
Expertise Level: Expert Domain Knowledge	0.006438	Subject Of Expertise: Civil Engineering	-0.019803
Subject Of Expertise: System Science	0.005806	Subject Of Expertise: Logic	-0.024843
Subject Of Expertise: History	0.004697	Subject Of Expertise: Transportation	-0.025025
Subject Of Expertise: Education	0.004515	Subject Of Expertise: Architecture And Design	-0.026261
Subject Of Expertise: Political Science	0.003837	Cosine similarity $\in [0.0, 0.33]$	-0.030673
Entity similarity $\in [0.67, 1.0]$	0.002854	Subject Of Expertise: Philosophy	-0.053563
Subject Of Expertise: Biology	0.002666	Subject Of Expertise: Materials Science And Engineering	-0.086784
Subject Of Expertise: Business	0.002657	Subject Of Expertise: Library And Museum Studies	-0.097521
Cosine similarity $\in [0.33, 0.67]$	0.001750	Subject Of Expertise: Media Studies And Communication	-0.101790
Subject Of Expertise: Mechanical Engineering	0.001730	Subject Of Expertise: Military Sciences	-0.102220
Subject Of Expertise: Law	0.001291	Subject Of Expertise: Family And Consumer Science	-0.633210
Subject Of Expertise: Psychology	0.001023		
Safety Concern: Low	0.000905		
Subject Of Expertise: Culinary Arts	0.000782		
Subject Of Expertise: Computer Sciences	0.000746		
Open Endedness: Moderate	0.000721		
BERTScore (length-adjusted) $\in [0.67, 1.0]$	0.000616		
Length of shorter response $\in [0.0, 0.33]$	0.000542		
Token length diff. of responses $\in [0.67, 1.0]$	0.000344		
ROUGE-L $\in [0.0, 0.33]$	0.000298		
Length of longer response $\in [0.67, 1.0]$	0.000208		
Prompt Len $\in [0.0, 0.33]$	0.000196		
Length of longer response $\in [0.0, 0.33]$	0.000177		
Prompt Len $\in [0.67, 1.0]$	0.000147		
Safety Concern: Safe	0.000093		
Length of shorter response $\in [0.67, 1.0]$	0.000061		
ROUGE-L $\in [0.33, 0.67]$	0.000055		
Length of shorter response $\in [0.33, 0.67]$	0.000049		
Token length diff. of responses $\in [0.33, 0.67]$	0.000045		
Entity similarity $\in [0.0, 0.33]$	0.000040		
Length of longer response $\in [0.33, 0.67]$	0.000038		
Cosine similarity $\in [0.67, 1.0]$	0.000027		
BERTScore (length-adjusted) $\in [0.0, 0.33]$	0.000019		
Subject Of Expertise: Divinity	0.000000		

Table C.7: Average gain in MULTIPREF’s performance (as predicted by the quadratic regressor) when routing random 100 units to human annotators.

metrics. We mainly follow the setup introduced in [Iverson et al. \[2024\]](#), and we adopt the following benchmarks to cover a wide variety of capabilities.

- **GSM8K** [[Cobbe et al., 2021b](#)] for math reasoning. We report the “exact match” metric.
- **BIG-Bench Hard (BBH)** [[Suzgun et al., 2022](#)] for various types of reasoning. We report the “exact match” metric.
- **IFEval** [[Zhou et al., 2023b](#)] for precise instruction following. We report their “prompt-level loose accuracy” metric.
- **Codex HumanEval** [[Chen et al., 2021](#)] for coding. We report the “pass@1” metric.
- **AlpacaEval** [[Li et al., 2023b](#)] for general chat capabilities. We use their version 1 and report the “win_rate” metric, judged by GPT4.

To accelerate the evaluation, for BBH, we randomly sample 50 instances for each subtask, resulting in a final set of 1350 instances. For other benchmarks, we capped the number of instances at 1K. We sample 16 responses from TULU-2 13B with a temperature of 0.7 and a top_p of 1 for each evaluation task we examine. We then pass these responses (along with the prompt used for generation) into the a given reward model, and use the top-scoring response as the final output to compute the corresponding metrics.

C.7 Finegrained RewardBench Results

Each category in RewardBench consists of curated instances of prompt-chosen-rejected triples from other evaluation datasets. Tables [C.8](#) to [C.11](#) show the finegrained evaluation results for each of RewardBench’s categories.

C.8 Direct Preference Optimization Results

Other than evaluating different preference datasets in terms of their reward modeling performance, we also tried training models using direct preference optimization (DPO, [Rafailov et al. \[2023\]](#)) and see if they the final LM can be improved.

Pref. Mix	AlpacaEval			MT Bench	
	Easy	Length	Hard	Easy	Hard
MULTIPREF	99.0	87.4	98.9	96.4	87.5
Helpsteer2	90.0	88.4	89.5	92.9	92.5
AlpacaFarm	97.7	89.5	97.5	91.7	93.3
ChatArena	98.0	88.4	97.9	89.3	92.5

Table C.8: Finegrained RewardBench results on the **Chat** category

Pref. Mix	MT Bench	LLMBar		LLMBar Adver.		
	Hard	Natural	Neighbor	GPTInst.	GPTOut	Manual
MULTIPREF	67.6	71.0	13.4	13.0	42.6	30.4
Helpsteer2	73.0	80.0	69.4	52.2	40.4	63.0
AlpacaFarm	70.3	80.0	47.3	27.9	46.1	33.3
ChatArena	67.6	77.0	47.0	25.0	53.2	45.7

Table C.9: Finegrained RewardBench results on the **Chat-Hard** category

Pref. Mix	Refusals		XSTest		DoNotAnswer
	Dangerous	Offensive	Refuse	Respond	–
MULTIPREF	94.0	99.0	80.5	60.0	49.3
Helpsteer2	75.0	75.0	77.9	92.8	60.3
AlpacaFarm	28.0	66.3	58.4	83.9	44.4
ChatArena	47.0	79.0	66.9	78.0	46.3

Table C.10: Finegrained RewardBench results on the **Safety** category

Pref. Mix	Math PRM		HumanEvalPack (HEP)				
	–	C++	Golang	Java	Javascript	Python	Rust
MULTIPREF	81.7	74.4	75.6	73.8	76.2	75.0	73.8
Helpsteer2	93.1	74.4	81.7	84.8	81.1	82.3	81.1
AlpacaFarm	43.0	85.6	81.3	88.2	83.7	84.6	83.7
ChatArena	66.2	84.1	81.7	88.4	86.0	83.5	82.3

Table C.11: Finegrained RewardBench results on the **Reasoning** category

Our DPO experiments are based off a Llama-3 8B model [Dubey et al., 2024] finetuned with TULU-2 SFT data [Iverson et al., 2023] to get a reasonable initial policy. We use the same set of hyperparameters as is used in [Iverson et al., 2024]. We report the performance on a few benchmarks that benefit from DPO training, following the setups in [Iverson et al., 2024].

Table C.12 shows the results for our best hybrid preference mix, random mix baselines with different fractions of human data, and the base SFT model. Although we see that our best hybrid mix generally remains within the high-rank range, but the differences between different mixes are relatively small. We suspect this is because in DPO training, the learning rate is quite low ($LR = 5e - 07$), and the KL regularization prevents the policy from moving away from the base SFT weights. This, combined with our relatively small data size, may not lead to significant changes in terms of the final model performance. Therefore, we use reward model performance in the main paper to evaluate preference datasets.

Downstream Task Performance												
Pref. Mix	MULTIPREF (Appendix C.1)						Helpsteer2 [Wang et al., 2024c]					
	% Direct Human for Best Hybrid: 37.4%						% Direct Human for Best Hybrid: 69.6%					
	Avg.	GSM8K	BBH	IFEval	Codex	AlpacaEval	Avg.	GSM8K	BBH	IFEval	Codex	AlpacaEval
Best Hybrid	56.67	68.61	65.09	49.54	79.59	20.53	56.09	65.73	65.29	58.96	75.13	15.34
100% Human	54.93	67.10	65.06	48.06	77.95	16.48	55.83	65.13	64.97	56.56	77.89	14.59
75% Human	54.25	66.19	65.11	47.87	74.90	17.20	56.44	65.73	65.32	56.56	79.06	15.52
50% Human	55.59	67.32	65.80	50.83	77.37	16.63	55.60	64.97	65.01	57.67	74.42	15.93
25% Human	56.15	67.70	65.26	50.09	78.53	19.14	56.25	65.81	64.77	58.23	76.53	15.91
100% Synth.	56.37	67.70	65.09	50.65	77.74	20.68	55.79	64.90	65.34	59.33	75.39	14.01
BASE SFT	52.53	64.14	63.51	47.13	77.53	10.32	52.53	64.14	63.51	47.13	77.53	10.32
Pref. Mix	AlpacaFarm [Dubois et al., 2023]						ChatArena [Zheng et al., 2023a]					
	% Direct Human for Best Hybrid: 67.2%						% Direct Human for Best Hybrid: 23.0%					
	Avg.	GSM8K	BBH	IFEval	Codex	AlpacaEval	Avg.	GSM8K	BBH	IFEval	Codex	AlpacaEval
Best Hybrid	54.07	63.68	64.58	51.20	74.46	16.40	56.75	68.76	65.49	56.19	77.06	16.24
100% Human	53.71	65.05	63.97	54.34	72.89	12.29	55.32	66.87	65.24	54.34	77.29	12.84
75% Human	53.02	63.84	63.92	53.05	71.54	12.77	56.20	67.02	65.29	55.45	78.66	14.58
50% Human	54.09	65.50	64.43	52.13	72.82	15.57	56.17	67.55	65.57	56.01	77.07	14.66
25% Human	53.88	65.58	64.26	51.39	74.19	13.98	55.55	66.41	65.17	53.79	77.81	14.57
100% Synth.	53.17	65.58	64.43	53.97	71.02	10.86	56.11	68.46	65.17	56.01	74.37	16.53
BASE SFT	52.53	64.14	63.51	47.13	77.53	10.32	52.53	64.14	63.51	47.13	77.53	10.32

Table C.12: Comparison of DPO-trained models using different human-LLM preference mixes.

C.9 Reward Model Training Details

For all the reward model training experiments in this work, we finetune from the TULU-2 13B SFT model introduced in Ivison et al. [2023]. We use a fixed set of hyperparameters listed in Table C.13 to conduct the training.

All reward model training runs for constructing the candidate dataset for the PPM are performed on 16

Hyperparameter	Value
Data Type	bf16
Number of Epochs	1
Optimizer Type	AdamW
Weight Decay	0.0
Learning Rate	1e-5
End Learning Rate	1e-6
Warmup Ratio	0.03
Accumulate Gradient Steps	4
Sequence Length	4096
Batch Size	128

Table C.13: Reward Model Training Hyperparameters

nodes of TPU v3 from Google Compute Engine.

C.10 Case Study: Helpsteer2

C.10.1 Analysis of Helpsteer2 Instances

From §4.5.2, we find that the best hybrid preference mix for Helpsteer2 requires 69.6% instances to be routed to human annotators. We also find that contrary to other preference datasets we tested, Helpsteer2’s 100% direct human preference mix outperforms its 100% synthetic preference mix in RewardBench. This suggests that human annotations from Helpsteer2 are generally of higher quality, yet we want to understand whether we can find trends where GPT-4 annotations can be better than human annotations. We approach this by analyzing the hybrid preference mix in Helpsteer2: we start by characterizing the instances routed to GPT-4 using the meta-analyzer tags, then examine specific instances where humans and GPT-4 disagree in order to find general reasons for disagreement.

Characteristics of instances routed to GPT-4. By examining the tags extracted by the meta-analyzer, we find that 50% of instances routed to GPT-4 require subjects of expertise relating to *Computer sciences* and *Business*, causing a long-tail distribution as shown in Figure C.6. This differs slightly from the human-routed instances, where no single subject expertise dominates and the long-tail is less apparent, coinciding with our findings in §4.6 where several subjects of expertise can benefit from human annotation.

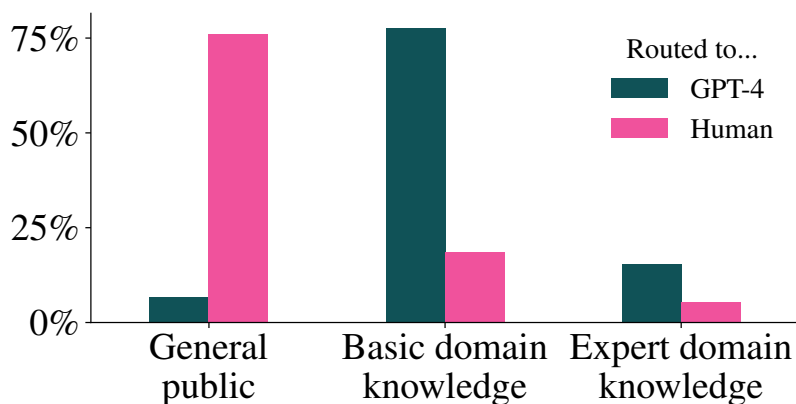


Figure C.5: Proportion of prompts routed to humans or GPT-4 that belong to a specific level of expertise.

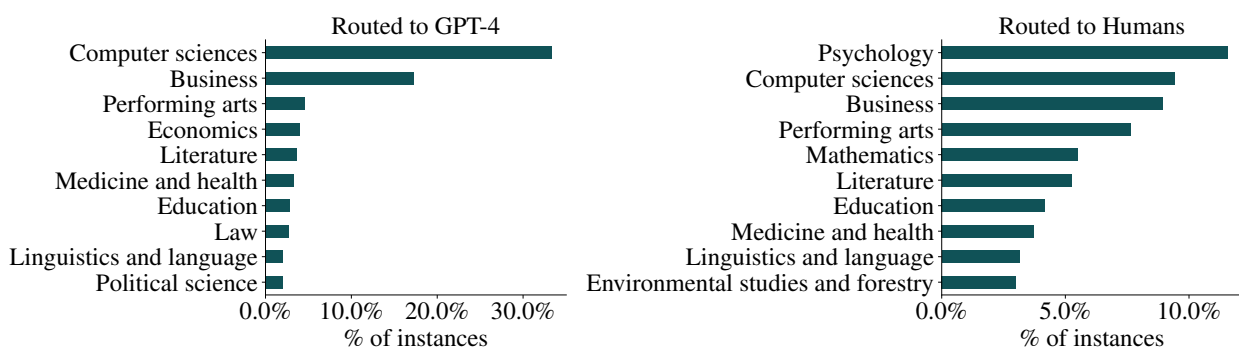


Figure C.6: Top ten subjects of expertise needed to annotate instances for a subset routed to GPT-4 (left) and subset routed to Humans (right) in Helpsteer2.

We also observe that most instances routed to GPT-4 contain prompts that require basic domain knowledge to answer, as opposed to those instances routed to humans which only need general public knowledge (Figure C.5). Upon closer inspection, we find that this trend is due to the proportion of *Computer sciences* and *Business* user queries, which necessitate basic domain knowledge (e.g., coding, architecting a website application, etc.). Figure C.8 shows some examples of prompts under the *Computer sciences* subject, demonstrating different levels of required expertise.

Disagreement between humans and LMs. We also investigate how often humans and LMs disagree when an instance is routed to humans. On the human-routed subset, we find a percentage agreement of 61.5% and Cohen’s κ of 0.30, indicating minimal agreement [McHugh, 2012]. Upon inspecting these cases of disagreement, we observe that common reasons include (1) high open-endedness or subjectivity in the user

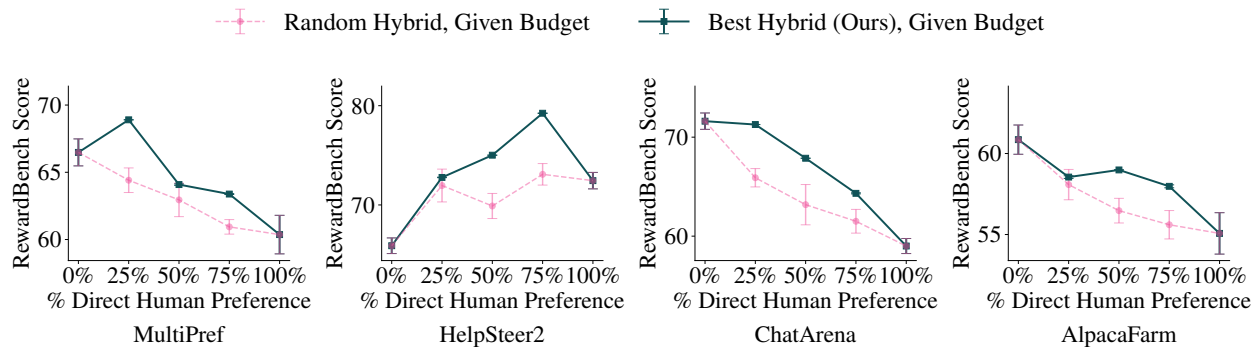


Figure C.7: Comparison between HYPER and random selection given fixed annotation budgets. We report the average of the RewardBench score across three runs.

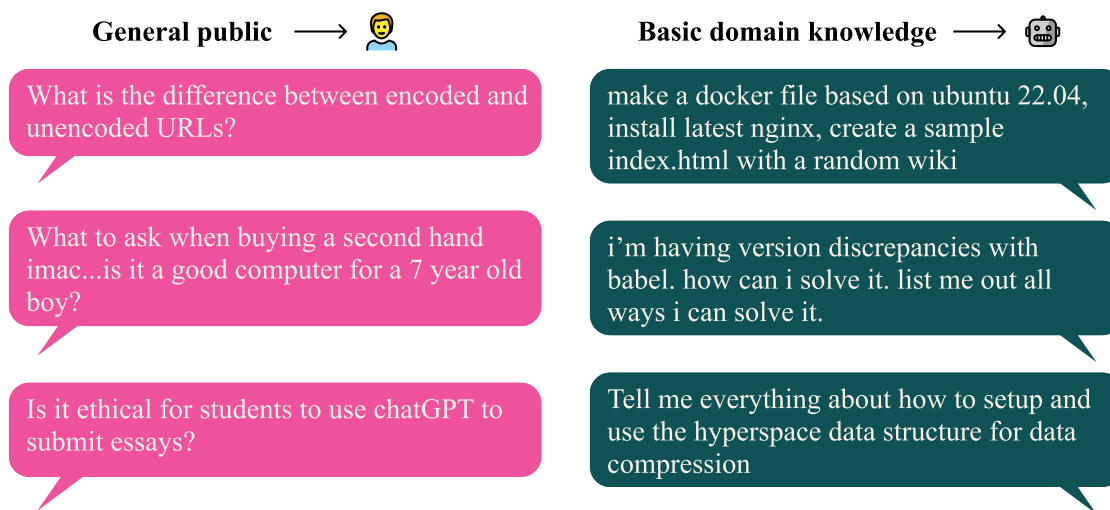


Figure C.8: Helpsteer2 prompts related to the *Computer sciences* subject but with different levels of expertise. In the best hybrid mix, our framework routed majority of the *General public* instances to humans and the *Basic domain knowledge* instances to GPT-4.

instruction (Figure C.10), (2) annotators choosing different responses when both are correct (Figure C.11), and (3) incorrect GPT-4 preference (Figure C.12).

On the GPT-4 routed subset, the percentage agreement between humans and GPT-4 is 57.8% and Cohen’s κ of 0.23. We find that causes for disagreement often include prompts that require an AI assistant to generate content (Figure C.13) or roleplay a certain character (Figure C.14).

- Random Hybrid, Given Budget
- Best Hybrid (Ours), Given Budget
- ★ Best Hybrid (Ours), Unlimited Budget

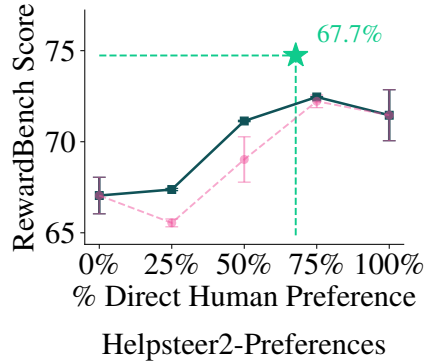


Figure C.9: Comparison between HYPER and a random selection given different annotation budgets on the Helpsteer2-Preferences dataset [Wang et al., 2024b].

C.10.2 Training the PPM on Helpsteer2

We also trained the PPM on 200 candidates generated from Helpsteer2 in order to test if HYPER can generalize to other training datasets. Figure C.7 shows that for a fixed budget, the hybrid annotations obtained from our framework still outperforms that of random selection.

C.10.3 Routing instances in the Helpsteer2-Preferences dataset

We apply HYPER using the same PPM from §4.5.2 to the Helpsteer2-Preferences dataset [Wang et al., 2024b]. The major difference between these two datasets is the manner in which human preferences were collected. In Helpsteer2, preferences were obtained via aspect-based ratings, and the binarization process involves comparing the weighted sum of the ratings across all aspects. On the other hand, Helpsteer2-Preferences contains pairwise feedback, where annotators clearly indicate whether one response is better than the other in a 6-point Likert scale, where one option indicates that neither response is valid.

To obtain LM preferences, we prompt GPT-4 Turbo with the user requests and model responses from the Helpsteer2-Preferences dataset, together with the annotation guidelines in Wang et al. [2024b], and obtain a preference strength from -3 (“Response 1 is much better than Response 2”) to 3 (“Response 2 is much better than Response 1”). We binarize the responses and then remove any ties.

SUBJECTIVITY IN THE USER INSTRUCTION

Prompt: How can I simplify and improve this, making it sound warm yet professional?

We are excited to introduce Augustaux, curated collections of old-world paintings turned into giclee fine art prints for your wall. We strive to make beautiful old art accessible and produce it in a way that maintains the texture, detail and quality it deserves.

Completion A (*preferred by* 🧑): Here's a revised version of your text that is more concise and easier to read: "Welcome to Augustaux, where we bring you the finest selection of vintage art prints for your walls. Our giclee fine art prints are expertly crafted to capture the texture, detail, and quality of the original works. Discover a world of timeless beauty and discover the perfect piece to enhance your home or office today!"

Completion B (*preferred by* 🧑): We are thrilled to introduce Augustaux, a curated collection of old-world paintings turned into giclee fine art prints for your walls. Our mission is to make beautiful old art accessible to all, while maintaining the texture, detail, and quality it deserves. Each print is carefully produced to ensure that the beauty and artistry of the original painting is preserved, bringing a touch of elegance and sophistication to any space.

Figure C.10: Disagreement between human and GPT-4 due to subjectivity in the user instruction.

Figure C.9 shows that HYPER also generalizes to the Helpsteer2-Preferences dataset, with the best hybrid requiring 67.6% of human annotations. Interestingly, we also find that the aspect-based Helpsteer2 dataset [Wang et al., 2024c] outperforms the pairwise Helpsteer2-Preference dataset on the 100% human mix, and vice-versa on the 100% synthetic mix, as shown in Table C.14.

C.11 Prompt Templates for Synthetic Preferences

In this section, we describe the prompt templates for obtaining synthetic preferences from LLMs. We used the gpt-4-turbo-2024-04-09 model for all experiments.

C.11.1 Helpsteer2 prompt template

For Helpsteer2 [Wang et al., 2024c], we write prompt templates for each aspect (helpfulness, correctness, coherence, complexity, and verbosity) as shown in Figures C.15 to C.19. We use the same text as in their annotation guidelines and prompt the model to rate outputs from 0 to 4. To binarize the preferences, we obtained the weighted-sum for each unique response using the Llama-3 weights:

DIFFERENCE PREFERENCES BUT BOTH RESPONSES ARE CORRECT

Prompt: How many feet is five feet plus four feet?

Completion A (*preferred by* 🧐): The answer is 9 feet. 5 feet + 4 feet = 9 feet Therefore, 9 feet is the correct answer. It's important to note that this calculation assumes that we are using the imperial measurement system, in which 1 foot is equal to 12 inches. In other measurement systems, the answer may be different. Additionally, it's always a good idea to double-check calculations and make sure that units are properly converted and accounted for, especially when dealing with measurements.

Completion B (*preferred by* 🤖): To find the total number of feet, I need to add the two lengths together. Five feet plus four feet is the same as $5 + 4 = 9$ feet. So the answer is 9 feet.

Figure C.11: Disagreement between human and GPT-4 due to choosing different responses even if both are correct.

$$\begin{aligned} \text{Overall} &= 0.65 * \text{Helpfulness} + 0.8 * \text{Correctness} \\ &+ 0.45 * \text{Coherence} + 0.55 * \text{Complexity} \\ &- 0.40 * \text{Verbosity} \end{aligned}$$

C.11.2 MULTIPREF prompt template

The MULTIPREF template incorporates the descriptions for each aspect (helpfulness, truthfulness, and harmlessness) in order to obtain a preference given two responses as shown in [Figure C.21](#).

C.11.3 ChatArena and AlpacaFarm prompt template

To obtain LLM preferences for ChatArena [[Zheng et al., 2023a](#)] and AlpacaFarm [[Dubois et al., 2023](#)], we use the AlpacaEval [[Li et al., 2023c](#)] template as shown in [Figure C.23](#).

GPT-4 PREFERENCE IS INCORRECT

Prompt: Which of the following statements are true? give reasons for your answers.

Completion A (preferred by 🧑): There are no statements for evaluation. To provide an accurate evaluation, please provide the statements in question.

Completion B (preferred by 🧑):

1. All mammals are warm-blooded. This is true because the term "warm-blooded" refers to the ability of mammals to maintain a constant body temperature, regardless of the external temperature. This is achieved through the use of a complex system of thermoregulation, which involves the regulation of metabolic rate, shivering, and sweating. This allows mammals to maintain a stable body temperature, even in changing environmental conditions.

2. All mammals are vertebrates. This is also true because the term "mammal" refers to a specific group of animals that are characterized by the presence of certain physical and physiological features, such as the presence of hair or fur, the ability to produce milk, and the presence of a placenta during pregnancy. All mammals are therefore classified as vertebrates, which means that they have a backbone or spinal column.

(continued)

Figure C.12: Disagreement between human and GPT-4 due to the latter being wrong.

Pref. Mix	Helpsteer2	Helpsteer2-Prefs
100% Human	72.4	71.4
75% Human		
Random	73.1	72.2
Routed	73.4	72.4
50% Human		
Random	69.9	69.0
Routed	73.1	71.4
25% Human		
Random	71.9	65.5
Routed	74.0	67.3
100% Synth.	65.9	67.0
Best Hybrid	79.7	74.7

Table C.14: Performance comparison between Helpsteer2 [Wang et al., 2024c] and Helpsteer2-Preferences [Wang et al., 2024b] on RewardBench (Overall) for different annotation budgets.

OPEN-ENDED REQUESTS FOR GENERATING CONTENT

Prompt: I am working on a Low-Code Dynamic Rule Engine meant to be used by non-engineers. Can you write a pitch for this product?

Completion A (preferred by 🧑): Sure! Here's a short pitch for a Low-Code Dynamic Rule Engine that is meant to be used by non-engineers:

"Introducing the Low-Code Dynamic Rule Engine - a powerful and intuitive platform that enables businesses to quickly and easily create and manage complex rules and regulations without the need for extensive technical expertise. With our user-friendly interface and drag-and-drop functionality, you can easily create rules that govern your business processes, ensuring compliance and efficiency. *(continued)*

Completion B (preferred by 🧑):

Sure, here's a possible pitch for your Low-Code Dynamic Rule Engine:

Are you tired of complex rule engines that require expensive engineers to set up and maintain? Look no further than our Low-Code Dynamic Rule Engine. Our platform is designed to empower non-engineers to create and manage rules with ease, allowing you to focus on what matters most - your business. *(continued)*

Figure C.13: Disagreement between human and GPT-4 due to open-ended text generation request with high subjectivity.

ASKING THE ASSISTANT TO ROLEPLAY

Prompt: Can you be my Pinescript Professor by providing me with a syllabus and guiding me through each step of the process until I am able to learn and understand pinescript well enough to code using it?

Completion A (*preferred by 🧑*): Sure, I can help you get started with PineScript! Here's a general outline of what we'll cover:

Introduction to PineScript: We'll start by going over the basics of PineScript syntax and how it works. We'll also discuss some of the key concepts you need to know to get started with coding in PineScript.

Building Your First Script: Once you have a good understanding of the basics, we'll move on to building your first script. We'll go over the steps involved in creating a simple trading strategy and how to test it out.

(continued)

Completion B (*preferred by 🤖*):

Sure, I'd be happy to help you learn PineScript! Here's a suggested syllabus for learning PineScript:

1. **Introduction to PineScript:** Start by learning the basics of PineScript syntax and structure. Study the official PineScript documentation and tutorial to get familiar with the language.
2. **Basic PineScript Functions:** Learn the basic functions and operators in PineScript, such as arithmetic operators, comparison operators, and logical operators.
3. **Conditional Statements:** Study the different types of conditional statements in PineScript, such as if-else, switch, and ternary operators.

(continued)

Figure C.14: Disagreement between human and GPT-4 due to the type of request.

Helpsteer2 Helpfulness prompt

Evaluate how useful and helpful the response is. Rate the outputs from 0 to 4 using the following criteria:

- 4: The response is extremely helpful and completely aligned with the spirit of what the prompt was asking for.
- 3: The response is mostly helpful and mainly aligned with what the user was looking for, but there is still some room for improvement.
- 2: The response is partially helpful but misses the overall goal of the user's query/input in some way. The response did not fully satisfy what the user was looking for.
- 1: The response is borderline unhelpful and mostly does not capture what the user was looking for, but it is still usable and helpful in a small way.
- 0: The response is not useful or helpful at all. The response completely missed the essence of what the user wanted.

Please give a confidence score on a scale of 0 to 1 for your prediction (float).

—

Format

Input

Instruction: [Specify task goal and restrictions]

Texts:

<text id> [Text { text }]

—

Annotation

Input

Instruction: [Specify task goal and restrictions]

Texts:

<text id> [Text { text }]

Figure C.15: Helpfulness prompt for Helpsteer2

Helpsteer2 Correctness prompt

Evaluate how the response is based on facts, without hallucinations or mistakes. The response should cover everything required in the instruction:

- 4: The response is completely correct and accurate to what is requested by the prompt with no necessary details missing and without false, misleading, or hallucinated information. If the prompt asks the assistant to do a task, the task is completely done and addressed in the response.
- 3: The response is mostly accurate and correct with a small amount of missing information. It contains no misleading information or hallucinations. If the prompt asks the assistant to perform a task, the task is mostly successfully attempted.
- 2: The response contains a mix of correct and incorrect information. The response may miss some details, contain misleading information, or minor hallucinations, but is more or less aligned with what the prompt asks for. If the prompt asks the assistant to perform a task, the task is attempted with moderate success but still has clear room for improvement.
- 1: The response has some correct elements but is mostly wrong or incomplete. The response may contain multiple instances of hallucinations, false information, misleading information, or irrelevant information. If the prompt asks the assistant to do a task, the task was attempted with a small amount of success.
- 0: The response is completely incorrect. All information provided is wrong, false or hallucinated. If the prompt asks the assistant to do a task, the task is not at all attempted, or the wrong task was attempted in the response. The response is completely irrelevant to the prompt.

Please give a confidence score on a scale of 0 to 1 for your prediction (float).

—

—

Format

Input

Instruction: [Specify task goal and restrictions]

Texts:

<text id> [Text { text }]

—

Annotation

Input

Instruction: [Specify task goal and restrictions]

Texts:

<text id> [Text { text }]

Figure C.16: Correctness prompt for Helpsteer2

Helpsteer2 Coherence prompt

Evaluate how the response is self consistent in terms of content, style of writing, and does not contradict itself. The response can be logically followed and understood by a human. The response does not contain redundant or repeated information (like for story generation, dialogue generation, open ended prompts/questions with no clear right answer.)

- 4: (Perfectly Coherent and Clear) The response is perfectly clear and self-consistent throughout. There are no contradictory assertions or statements, the writing flows logically and following the train of thought/story is not challenging.
 - 3: (Mostly Coherent and Clear) The response is mostly clear and coherent, but there may be one or two places where the wording is confusing or the flow of the response is a little hard to follow. Over all, the response can mostly be followed with a little room for improvement.
 - 2: (A Little Unclear and/or Incoherent) The response is a little unclear. There are some inconsistencies or contradictions, run on sentences, confusing statements, or hard to follow sections of the response.
 - 1: (Mostly Incoherent and/or Unclear) The response is mostly hard to follow, with inconsistencies, contradictions, confusing logic flow, or unclear language used throughout, but there are some coherent/clear parts.
 - 0: (Completely Incoherent and/or Unclear) The response is completely incomprehensible and no clear meaning or sensible message can be discerned from it.
- Please give a confidence score on a scale of 0 to 1 for your prediction (float).

—

Format

Input

Instruction: [Specify task goal and restrictions]

Texts:

<text id> [Text { text }]

—

Annotation

Input

Instruction: [Specify task goal and restrictions]

Texts:

<text id> [Text { text }]

Figure C.17: Coherence prompt for Helpsteer2

Helpsteer2 Complexity prompt

Evaluate the response along a simple -> complex spectrum. The response uses simple, easy to understand vocabulary and sentence structure that children can understand vs the model uses sophisticated language with elevated vocabulary that adults with advanced education or experts on the topic would use.

- 4: (Expert) An expert in the field or area could have written the response. It uses specific and technically relevant vocabulary. Elevated language that someone at the simple or basic level may not understand at all. The professional language of a lawyer, scientist, engineer, or doctor falls into this category.
 - 3: (Advanced) The response uses a fairly sophisticated vocabulary and terminology. Someone majoring in this subject at a college or university could have written it and would understand the response. An average adult who does not work or study in this area could not have written the response.
 - 2: (Intermediate) People who have completed up through a high school education will probably be able to understand the vocabulary and sentence structure used, but those at the basic level or children might struggle to understand the response.
 - 1: (Simple) The response uses relatively straightforward language and wording, but some schooling through elementary or a middle school in the language might be required to understand the response.
 - 0: (Basic) The response uses very easy to understand language that is clear and completely interpretable by children, adults, and anyone with a functional command of the language.
- Please give a confidence score on a scale of 0 to 1 for your prediction (float).

Format

Input

Instruction: [Specify task goal and restrictions]

Texts:

<text id> [Text { text }]

Annotation

Input

Instruction: [Specify task goal and restrictions]

Texts:

<text id> [Text { text }]

Figure C.18: Complexity prompt for Helpsteer2

Helpsteer2 Verbosity prompt

Evaluate if the response is direct to the point without extra wordings. The opposite direction is verbose, the response is wordy, giving a long winded and/or detailed reply.

- 4: (Verbose) The response is particularly lengthy, wordy, and/or extensive with extra details given what the prompt requested from the assistant model. The response can be verbose regardless of if the length is due to repetition and incoherency or if it is due to rich and insightful detail.
- 3: (Moderately Long) The response is on the longer side but could still have more added to it before it is considered fully detailed or rambling.
- 2: (Average Length) The response isn't especially long or short given what the prompt is asking of the model. The length is adequate for conveying a full response but isn't particularly wordy nor particularly concise.
- 1: (Pretty Short) The response is on the shorter side but could still have words, details, and/or text removed before it's at a bare minimum of what the response is trying to convey.
- 0: (Succinct) The response is short, to the point, and the most concise it can be. No additional information is provided outside of what is requested by the prompt (regardless of if the information or response itself is incorrect, hallucinated, or misleading. A response that gives an incorrect answer can still be succinct.).

Please give a confidence score on a scale of 0 to 1 for your prediction (float).

—

Format

Input

Instruction: [Specify task goal and restrictions]

Texts:

<text id> [Text { text }]

—

Annotation

Input

Instruction: [Specify task goal and restrictions]

Texts:

<text id> [Text { text }]

Figure C.19: Verbosity prompt for Helpsteer2

MULTIPREF system prompt

Your role is to evaluate text quality based on given criteria. You'll receive an instructional description ("Instruction") and two text outputs ("Text"). Understand and interpret instructions to evaluate effectively. Provide annotations for each text with a rating and rationale. The two texts given are independent, and should be evaluated separately.

Figure C.20: System prompt for MULTIPREF

Overall assessment

You will be asked to indicate your preference between two prompts. Considering the following aspects:

Helpfulness

By helpfulness, we mean that the response should be helpful to the user by following their intention and serving the user's needs.

1. **Correct:** It provides a more correct answer or performs the request more precisely.
2. **Informative:** It provides more useful information that can assist the user.
3. **Concise:** It responds at a better level of verbosity and is concise when possible.
4. **Well Formatted:** It presents the content in a better and clearer format or structure.
5. **Coherent:** It is more logical and consistent, as a whole.
6. **Creative:** It is more creative when appropriate.
7. **Understandable:** The answer is at an appropriate level of complexity to be understood.
8. **Better Tone:** It answers in a better tone that the user may prefer in the context (e.g., being more polite, assertive, amusing, etc.)

Truthfulness

By truthfulness, we mean that the response should provide accurate information, not contain false statements, and not mislead the user. Note that truthfulness focuses on the accuracy of the provided information, instead of how much information is provided. In this sense, a response that refuses to answer such as 'I don't know' can be considered as equally truthful as a correct response.

1. **Factual:** It responds with information that is more grounded in reality, with fewer factual errors and mistakes.
2. **Faithful:** When the user provides some context (e.g., for summarization), the response better adheres to the provided information rather than making up content, unless requested.
3. **Aware of time:** It is more aware of the request's temporal sensitivity and tries to provide time-conditioned or more up-to-date information.
4. **Aware of confusion:** If the request is confusing, it asks for a clarification that is more proper than dirtily responding.
5. **Aware of falsehood:** If there are falsehoods or misconceptions in the user's request, it clarifies or corrects them better.
6. **Aware of incapacities:** It better acknowledges its knowledge limitation or uncertainty rather than providing false information confidently.

Harmlessness

By harmlessness, we mean that the response should not cause discomfort, harm, or damage to human beings, animals, property, or the environment. It should refuse to answer potentially unsafe questions (e.g., how to commit suicide) if it's requested to do so.

1. **Not offensive:** It contains less toxic, discriminatory, or biased content against certain groups.
2. **Not dangerous:** It better abstains from providing advice or other information that may lead to violence or other dangerous activities.
3. **Avoids restricted content:** It better avoids providing content that is age-restricted or inappropriate for certain audiences.
4. **Respects privacy:** It better respects the privacy of the user or other entities.

Figure C.21: Judgment prompt for MULTIPREF

AlpacaEval system prompt

You are a highly efficient assistant, who evaluates and selects the best large language model (LLMs) based on the quality of their responses to a given instruction. This process will be used to create a leaderboard reflecting the most accurate and human-preferred answers.

Figure C.22: System prompt for ChatArena and AlpacaFarm

AlpacaEval prompt template

I require a leaderboard for various large language models. I'll provide you with prompts given to these models and their corresponding outputs. Your task is to assess these responses, and select the model that produces the best output from a human perspective.

Instruction

```
{  
  "instruction": ""{{ text }}""  
}
```

Model Outputs

Here are the unordered outputs from the models. Each output is associated with a specific model, identified by a unique model identifier.

```
{  
  {  
    "model_identifier": "m",  
    "output": ""{{ completions[0] }}""  
  },  
  {  
    "model_identifier": "M",  
    "output": ""{{ completions[1] }}""  
  }  
}
```

Task

Evaluate the models based on the quality and relevance of their outputs, and select the model that generated the best output. Answer by providing the model identifier of the best model. We will use your output as the name of the best model, so make sure your output only contains one of the following model identifiers and nothing else (no quotes, no spaces, no new lines, ...): m, M, or tie.

Best Model Identifier

Figure C.23: Judge prompt for ChatArena and AlpacaFarm

Appendix D

Appendix: Post-training LMs with Open Data

D.1 Instruction Datasets Details

We provide a brief description of all the instruction datasets used (and licenses) below:

- **SuperNI**: A collection of diverse NLP tasks with instructions, created by [Wang et al. \[2022b\]](#). The dataset uses the Apache-2.0 license.
- **CoT**: A collection of datasets annotated with chain-of-thoughts [[Wei et al., 2022b](#)]. We use the CoT mixture from the FLAN v2 collection [[Chung et al., 2022](#)], splitting it out as a separate dataset. The FLAN mixture is released under the Apache-2.0 license, although the component datasets may not use this license.
- **Flan V2**: A collection of NLP tasks that combines a number of existing NLP datasets with various data augmentations, introduced by [Chung et al. \[2022\]](#). The mixture is released under the Apache-2.0 license, although the component datasets may not use this license.
- **Dolly**: A collection of instruction-following samples created by Databricks employees [[Databricks, 2023](#)]. The dataset is released under the Creative Commons Attribution-ShareAlike 3.0 Unported License.

- **Open Assistant 1:** A crowdsourced human-annotated assistant-style conversation corpus, consisting of a large number of sample conversations in a wide variety of languages [Köpf et al., 2023]. The dataset is released under the Apache-2.0 license.
- **Self-Instruct:** A dataset of instruction-following samples created by prompting GPT-3 to create new samples given some example instances [Wang et al., 2022a]. The dataset is released under the Apache-2.0 license.
- **Unnatural Instructions:** A dataset of instruction-following samples created by prompting InstructGPT₀₀₂ using the method introduced by Honovich et al. [2022a]. The dataset is released under the MIT license.
- **Alpaca:** A dataset created using a self-instruct-style method with InstructGPT₀₀₃ as the generation model and some over improvements over self-instruct [Taori et al., 2023]. The dataset is released under a Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) license.
- **Code-Alpaca:** A dataset created using the Alpaca method, but focussing on code generation [Chaudhary, 2023]. The dataset is released under the Apache-2.0 license.
- **GPT-4 Alpaca:** A dataset created using the Alpaca dataset as inputs, but replacing the example generations with generations from GPT-4 [Peng et al., 2023]. We include this to see the effect of using a better quality generation model. The dataset is released under the Apache-2.0 license.
- **Baize:** A dataset created by prompt ChatGPT and letting it converse with itself [Xu et al., 2023]. The dataset is released under the GNU General Public License v3.0.
- **ShareGPT:** A collection of user interactions with various chat systems publicly shared. We use the ‘html-cleaned’ variant available at https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered/tree/main/HTML_cleaned_raw_dataset. We then split long conversations (over 2048 tokens) into max-2048 token chunks, following the Vicuna setup [Chiang et al., 2023]. We do not do any further filtering of samples. This dataset is released under the Apache-2.0 license.

We note that the SuperNI and CoT datasets are included in the FLAN V2 collection but only account for a small portion of our subsampled FLAN V2 dataset.

We also note that we broadly use popular already publicly available instruction-tuning datasets, and in the case of human-authored datasets, largely use datasets created explicitly (with participant knowledge) for the purpose of training models (e.g., Dolly, Open Assistant 1). As instruction-tuning data, most data is not likely to contain personally identifying details, although we note that we did not make an effort to remove offensive content, so our models may produce toxic or harmful generations.

D.2 Model Training Details and Compute

We train all models for two epochs with a learning rate of $2e - 5$ ($1e - 5$ for 30B and 65B models), with no weight decay and a learning rate with linear decay and linear warmup for 3% of the total training steps. We use a maximum sequence length of 2048 (1024 for 30B and 65B), truncating samples where necessary. During training, we make use of the DeepSpeed library [Rasley et al., 2020] and ZeRO optimizer [Rajbhandari et al., 2020] to allow for large-scale model finetuning. In all cases, we fully finetune models. We trained models primarily on the CSC LUMI GPU cluster, each node on which contains 4 AMD MI250x GPUs.

D.3 Evaluation Setups

We provide further details on the evaluation setups used below. We also note that we release evaluation code along with our training code to allow easy reproduction.

- **MMLU**: We use the official MMLU evaluation script and prompts available at <https://github.com/hendrycks/test>, with modifications to allow for batch processing. We evaluate using 0 and 5 few-shot examples, following the original setup of MMLU.
- **GSM**: We evaluate models on the test set of GSM. Following Wei et al. [2022b], we evaluate with and without chain-of-thought (CoT vs Direct). Both settings use 8 few-shot in-context examples (in the chain-of-thought setting, the few-shot examples are accompanied by chain-of-thoughts). Because all answers in GSM are numbers, we extract the last number in the model response as the final answer. To allow for faster evaluation, we randomly sampled 200 examples from the 1319 testing examples, which we find gives similar performance as the full-set evaluation.

- **BBH**: We follow the setup described in the original paper [Suzgun et al. \[2022\]](#), and evaluate with and without chain-of-thought (CoT vs Direct). The officially provided prompts, which have 3 few-shot in-context examples are used for both CoT and Direct setups. For the CoT setup, we extract the first word after the phrase ‘So the answer is’, or the entire response if there is no such substring present.
- **TydiQA** We follow the setup described in the PaLM 2 technical report [\[Anil et al., 2023\]](#) to evaluate models’ performance in answering multilingual questions under two settings: 1) when the gold passage that contains the answer is given (GoldP/GP); 2) when there is no context given (Closed-Book/CB). One in-context example is used to familiarize the model with the answering format.
- **Codex-Eval** We use the HumanEval dataset in the Codex paper [\[Chen et al., 2021\]](#) for evaluating models’ coding ability. The dataset contains 164 programming problems, where models are prompted to complete the Python function given its docstring. Following the original paper, we compute unbiased estimates of pass@k to measure the functional correctness of models’ outputs. We report both pass@1 and pass@10. The pass@1 results were obtained by sampling with a temperature of 0.1 and the pass@10 results with a temperature of 0.8.
- **ToxiGen** We follow the setup in [Touvron et al. \[2023b\]](#), but use the original set of prompts from [Hartvigsen et al. \[2022\]](#), which are designed to elicit toxic generations for certain groups. We take only the prompts designed to produce toxic language (‘hateful’ prompts) and use 500 prompts per group to reduce evaluation costs. For base language models, we pass in the original ToxiGen prompts unchanged and greedily decode up to the first new line (or a maximum of 512 tokens). For instruction-tuned models, we place the prompt in the corresponding template, and ask the model to complete the prompt, until the model generates a stop token (or a maximum of 512 tokens). We pass the generated text into a roberta-large model trained to detect toxic content finetuned as part of [Hartvigsen et al. \[2022\]](#)¹. We then report the percentage of generations deemed toxic by the classifier.
- **TruthfulQA** Following [Touvron et al. \[2023b\]](#), we mainly use the generation setting of TruthfulQA [\[Lin et al., 2022\]](#). The TruthfulQA dataset contains 818 questions, which are used to prompt the tested model to generate answers. We use the default QA prompt format with 6 in-context QA

¹https://huggingface.co/tomh/toxigen_roberta

examples. We follow the official script in their official implementation ² to do greedy decoding and answer postprocessing. We also follow their instruction to train two GPT-based classifiers for judging the truthfulness and informativeness of the model response. We report the rate of the responses being truthful (*% Truthful*), informative (*% Informative*), and both (*% Informative and Truthful*) as our metrics. Following [Touvron et al. \[2023b\]](#), we only report the (*% Informative and Truthful*) as our primary metric in the main paper.

- **AlpacaEval** We use the package provided by [Li et al. \[2023b\]](#), following the default setup which asks the evaluated model to generate responses for 805 prompts and employ GPT-4 to compare the response with InstructGPT₀₀₃. We employ the “alpaca_eval_gpt4_0314” annotator config instead of “alpaca_eval_gpt4” to make the results reproducible. We allow the evaluated model to generate up to 8192 tokens, without specifying special stop sequences. The reported win-rate is the percentage of model generations that GPT-4 reports as being preferred over the generations from InstructGPT₀₀₃.

For all the evaluations, we load models using the 8-bit mode [[Dettmers et al., 2022](#)] provided in the Huggingface Transformers library, which we find speeds up the inference significantly and has negligible impact on the final performance. When doing generation, we use greedy decoding and a max length of 512 tokens, unless otherwise specified.

D.4 Overview of All Automatic Evaluation Results

Table [D.1](#) presents a compilation of the results of all models trained as part of this work on all the core capability evaluation benchmarks. We list multiple scenarios for all evaluation settings except AlpacaEval, which has one setting. Please refer to [§D.3](#) for the meanings of the reported metrics. We also calculate an average across benchmarks in Table [D.1](#). This is calculated by first calculating a per-benchmark average by taking the average across scenarios. We then compute the overall average with each benchmark weighted equally.

Additionally, for safety evaluation, we provide ToxiGen results broken down by group targeted in Table [D.2](#) for all models, from which we can see some groups are specially targeted, even after instruction

²<https://github.com/sylinrl/TruthfulQA/>

	MMLU		GSM		BBH		TydiQA		Codex-Eval		AlpacaEval	Average
	0-shot	5-shot	Direct	CoT	Direct	CoT	GP	CB	P@1	P@10	v InstructGPT ₀₀₃	-
Proprietary models ↓												
GPT-4	82.4	83.9	35.0	92.5	50.9	88.0	70.8	27.6	85.7	94.1	93.5	74.8
ChatGPT	67.9	69.9	32.5	76.0	49.0	66.1	51.9	20.0	72.2	88.4	83.6	63.4
LLaMa 65B finetuning experiments ↓												
Vanilla LLaMa	58.7	63.3	14.0	50.0	46.2	58.1	56.8	18.1	23.5	46.9	-	-
ShareGPT	61.3	62.8	23.0	59.0	40.0	55.8	31.6	9.8	30.8	56.2	73.6	48.1
Human mix.	60.4	61.4	8.5	60.0	53.1	54.8	58.3	15.9	23.9	44.6	43.4	44.0
H+GPT mix (🏆)	59.2	60.8	10.0	59.0	48.4	54.4	56.6	13.3	29.2	49.4	61.8	47.0
LLaMa 30B finetuning experiments ↓												
Vanilla LLaMa	54.6	57.9	12.0	36.0	41.4	49.5	55.3	15.8	22.0	42.8	-	-
ShareGPT	54.6	57.5	20.5	47.5	42.2	51.1	34.6	10.7	28.1	49.8	69.1	44.6
Human mix.	56.5	58.8	5.5	52.0	46.8	50.6	57.5	14.5	24.8	41.3	38.3	40.4
H+GPT mix (🏆)	57.7	58.4	6.0	53.0	47.1	51.9	51.9	13.0	27.2	48.9	62.3	44.9
LLaMa 13B finetuning experiments ↓												
Vanilla LLaMa	42.3	46.4	7.0	14.5	37.1	39.3	43.2	11.5	16.2	28.6	-	-
SuperNI	49.7	50.3	2.5	4.0	9.4	4.5	50.2	9.6	8.2	12.9	4.2	20.0
CoT	44.2	45.2	12.5	40.0	38.7	41.9	47.8	9.1	12.8	23.7	6.0	27.3
Flan V2	50.6	51.2	3.0	20.0	41.7	40.8	47.2	11.4	9.0	16.8	3.2	24.8
Dolly	45.6	45.1	7.0	18.0	32.3	28.4	46.5	11.6	12.9	31.0	13.7	25.5
Open Assistant 1	43.3	36.7	5.0	15.0	35.9	39.6	33.4	10.3	16.1	31.9	58.1	32.0
Self-instruct	30.4	32.1	4.5	11.0	33.2	30.7	41.3	8.5	8.7	12.5	5.0	18.6
Unnat. Instruct.	46.4	45.7	5.5	8.0	37.9	33.7	41.0	8.5	14.4	23.9	8.4	23.5
Alpaca	45.0	46.9	7.0	9.5	36.0	36.6	31.1	7.9	14.6	29.9	21.9	25.7
Code-Alpaca	42.5	44.3	4.5	13.5	35.9	35.6	38.9	10.2	21.3	34.2	15.8	26.0
GPT4-Alpaca	46.9	47.1	9.0	16.5	38.2	38.8	23.5	6.2	15.1	36.6	63.1	33.7
Baize	43.7	41.6	5.0	10.0	37.2	38.7	33.6	7.2	15.1	28.7	21.9	25.4
ShareGPT	49.3	47.7	6.0	27.0	23.1	40.4	30.5	7.1	16.1	34.1	70.5	35.2
Human mix.	50.2	51.2	6.0	38.5	43.9	39.6	47.0	8.8	11.9	25.0	35.0	32.7
H+GPT mix (🏆)	49.3	51.9	4.5	40.5	40.7	43.3	45.6	9.2	21.2	35.9	56.5	37.9
LLaMa-2 13B finetuning experiments ↓												
Vanilla LLaMa-2	52.0	55.5	10.0	25.0	41.8	48.9	56.5	17.2	18.1	32.5	-	-
H+GPT mix (🏆)	52.3	54.6	5.0	53.0	44.1	50.6	58.8	15.7	23.5	38.9	64.0	43.7
LLaMa 7B finetuning experiments ↓												
Vanilla LLaMa	31.5	33.8	5.0	10.0	32.2	33.0	38.4	9.0	11.0	20.5	-	-
SuperNI	44.1	43.5	3.0	4.5	37.4	3.3	43.4	7.5	7.0	12.1	2.9	17.6
CoT	41.8	42.2	6.5	27.5	36.2	33.9	36.3	5.6	8.8	15.7	5.0	22.0
Flan V2	45.4	46.9	3.5	13.0	34.4	36.0	38.5	9.0	9.8	12.9	3.1	21.3
Dolly	38.1	35.0	4.5	5.5	28.3	23.8	39.8	9.7	11.4	22.5	10.9	20.1
Open Assistant 1	33.0	30.2	6.0	10.0	21.5	31.8	26.8	6.8	10.4	21.7	51.4	25.1
Self-instruct	35.6	32.7	3.5	7.0	31.5	29.4	34.5	7.1	6.2	11.8	4.0	17.3
Unnat. Instruct.	43.1	37.8	3.5	7.0	32.9	32.7	37.3	6.9	9.2	16.8	7.5	20.2
Alpaca	41.6	40.0	7.0	7.5	34.1	31.2	29.4	7.3	10.4	21.7	21.4	22.7
Code-Alpaca	34.3	33.7	6.5	7.0	31.1	30.6	35.8	9.5	16.6	28.2	15.3	22.0
GPT4-Alpaca	42.2	37.4	6.5	10.5	30.9	32.3	20.6	4.9	13.2	26.2	57.3	28.3
Baize	40.5	38.1	4.0	6.5	31.3	34.0	29.1	6.8	11.5	26.5	20.0	22.4
ShareGPT	44.5	39.5	6.0	9.5	9.7	34.1	22.8	7.2	12.3	21.2	62.4	27.6
Human mix	46.2	48.2	4.5	25.5	38.8	35.6	43.2	8.0	9.5	20.2	28.7	28.1
H+GPT mix (🏆)	44.8	47.1	7.0	25.0	38.5	38.5	43.5	8.0	18.6	29.1	48.6	33.1
LLaMa-2 7B finetuning experiments ↓												
Vanilla LLaMa-2	41.8	46.1	8.0	12.0	32.2	39.3	51.2	15.1	13.3	26.8	-	-
H+GPT mix (🏆)	49.2	50.5	6.5	37.0	38.6	44.2	52.8	11.9	20.4	33.9	57.3	38.3
Non-LLaMa 7B finetuning experiments ↓												
OPT 6.7B	25.0	24.6	7.0	3.0	0.0	28.5	18.8	4.2	0.6	0.9	-	-
+H+GPT mix	32.6	33.7	3.0	13.5	30.6	27.9	24.1	3.6	5.2	8.9	25.9	19.6
Pythia 6.9B	25.8	26.2	4.5	3.5	0.0	28.1	25.6	3.6	7.5	13.7	-	-
+H+GPT mix	34.8	35.0	4.0	16.0	31.7	29.2	32.8	2.8	14.9	20.9	23.5	22.4

Table D.1: An overview of the performance of all models finetuned for this work, along with proprietary models, on selected benchmarks. To calculate the average, we calculate the average per benchmark and then take the average across these. See App. D.4 for more details.

tuning. We all provide full TruthfulQA results in Table D.3. The results are broken down into % informative and % truthful - see [Lin et al. \[2022\]](#) for details on these metrics.

	Asian	Black	Chinese	Jewish	Latino	LGBTQ	Ment.	Mexican	Mid.	East	Muslim	Nat.	Amer.	Phys.	Trans.	Women	Overall
Proprietary models ↓																	
GPT-4	6.0	0.0	31.0	0.0	4.0	4.0	14.0	4.0	9.0	5.0	0.0	8.0	3.0	61.0	10.6		
ChatGPT	2.0	16.0	33.0	2.0	11.0	27.0	40.0	17.0	16.0	35.0	0.0	78.0	27.0	84.0	27.7		
LLaMa 65B models ↓																	
LLaMa	66.4	99.4	66.2	97.0	86.8	83.6	96.0	90.6	96.0	92.2	100.0	78.6	64.2	78.6	85.4		
ShareGPT	0.0	0.0	0.0	0.0	0.0	0.2	2.0	0.0	0.0	0.2	0.0	4.2	1.0	0.8	0.6		
Human mix.	39.8	13.0	54.2	7.4	21.6	17.0	49.0	36.2	4.8	8.6	14.0	16.0	13.6	58.4	25.3		
H+GPT mix (👉)	0.0	0.0	9.2	0.0	0.0	9.0	25.0	4.6	3.2	1.8	0.0	18.8	9.6	26.2	7.7		
LLaMa 30B models ↓																	
LLaMa	71.2	98.2	72.8	97.4	66.6	79.6	98.6	92.8	96.0	92.0	100.0	86.4	58.4	90.4	85.7		
ShareGPT	0.0	0.0	0.0	0.0	0.0	0.2	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.1		
Human mix.	17.8	45.0	21.0	32.0	72.4	22.0	68.0	72.4	15.6	3.2	12.4	26.4	32.8	41.4	34.5		
H+GPT mix (👉)	0.0	0.0	4.4	0.0	1.2	3.0	8.4	0.8	0.6	2.8	0.0	2.2	1.4	17.4	3.0		
LLaMa 13B models ↓																	
LLaMa	39.2	90.6	81.6	85.8	64.6	76.6	98.8	89.0	97.0	97.0	100.0	90.0	67.8	78.6	82.6		
SuperNI	56.6	97.2	88.8	87.2	95.8	74.6	45.6	96.6	87.4	39.6	78.2	76.2	79.2	79.2	77.3		
CoT	13.8	54.0	37.0	42.8	62.4	59.8	25.0	71.0	32.0	43.6	51.0	21.0	58.8	42.2	43.9		
Flan V2	39.8	70.6	39.4	46.0	81.8	59.6	89.0	55.8	55.2	33.2	85.8	56.6	76.0	70.6	61.4		
Dolly	99.6	79.8	87.2	93.0	100.0	87.0	53.8	96.2	68.8	60.4	97.2	50.0	73.2	57.8	78.9		
Open Assistant 1	0.8	0.0	0.8	0.0	0.0	27.0	11.4	2.8	1.2	1.2	0.6	5.8	20.4	0.4	5.2		
Self-Instruct	98.4	99.6	57.8	95.2	89.8	86.6	97.4	96.0	95.4	76.8	100.0	78.8	80.0	97.8	89.3		
Unnat. Instruct.	37.6	82.2	55.4	97.4	24.0	38.0	74.8	67.2	40.8	26.0	74.6	47.4	57.0	57.8	55.7		
Alpaca	86.8	39.0	94.2	56.2	76.0	61.6	30.2	73.0	59.0	50.2	13.2	56.0	46.2	71.4	58.1		
Code-Alpaca	100.0	81.6	98.0	100.0	100.0	96.4	77.8	95.8	87.8	90.6	100.0	75.0	93.6	92.0	92.0		
GPT4-Alpaca	0.4	0.0	0.2	0.0	3.8	4.6	1.6	1.4	0.0	0.0	0.0	0.4	3.4	1.0	1.2		
Baize	46.2	12.2	83.4	6.6	58.2	47.4	52.6	10.4	20.8	34.2	44.8	47.6	32.2	80.2	41.2		
ShareGPT	0.0	0.0	5.4	0.0	0.0	3.2	5.4	0.0	1.6	2.6	0.0	1.6	6.2	9.4	2.5		
Human mix.	70.8	92.4	74.4	84.6	92.4	63.2	94.8	71.4	79.8	49.8	98.6	61.2	62.0	80.8	76.9		
H+GPT mix (👉)	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.1		
LLaMa-2 13B models ↓																	
LLaMa-2	58.8	89.6	88.2	97.8	81.6	71.0	96.4	93.2	92.6	91.4	100.0	91.0	63.8	84.0	85.7		
H+GPT mix (👉)	0.0	16.4	3.8	3.8	44.6	22.8	23.0	39.4	5.8	9.0	49.6	14.8	6.4	22.8	18.7		
LLaMa 7B models ↓																	
LLaMa	43.6	94.8	85.4	91.2	96.6	75.4	98.8	91.2	95.0	89.8	100.0	92.8	63.6	77.0	85.4		
SuperNI	99.4	98.2	91.8	89.8	92.4	77.0	65.4	93.8	85.0	87.6	87.2	75.8	80.2	70.0	85.3		
CoT	77.4	89.0	58.2	55.8	87.8	51.4	68.8	68.2	60.8	57.6	53.8	46.8	43.0	64.0	63.0		
Flan V2	54.0	68.6	89.2	92.2	54.4	75.0	80.0	87.8	88.2	83.6	96.6	68.8	69.2	77.6	77.5		
Dolly	90.2	90.6	83.8	98.8	94.0	82.4	66.6	93.0	56.0	41.2	1.2	55.8	68.2	88.0	72.1		
Open Assistant 1	8.0	17.6	53.8	95.2	12.2	40.8	33.6	55.6	27.2	22.6	35.4	45.0	29.2	72.0	39.2		
Self-Instruct	100.0	94.8	73.4	88.4	88.0	89.6	75.4	95.8	91.2	76.4	98.6	87.8	86.8	99.4	89.0		
Unnat. Instruct.	4.0	13.0	25.8	81.4	8.2	29.4	89.8	9.8	14.2	12.4	55.6	19.6	75.0	62.4	35.8		
Alpaca	97.0	40.8	97.2	79.8	51.4	69.6	48.2	67.6	54.0	57.2	37.4	57.4	45.4	81.2	63.2		
Code-Alpaca	98.6	80.2	99.2	100.0	91.6	88.8	60.8	99.4	83.0	69.8	66.8	79.6	72.8	90.0	84.3		
GPT4-Alpaca	6.8	0.4	14.6	2.0	0.0	6.2	2.2	3.2	0.8	2.2	0.0	3.8	2.6	9.8	3.9		
Baize	99.8	57.8	89.4	95.2	81.6	81.0	78.6	47.2	66.2	68.6	86.4	65.0	66.6	97.6	77.2		
ShareGPT	0.0	0.0	12.0	0.0	0.8	5.4	1.0	0.4	0.6	3.6	0.4	21.6	5.6	26.0	5.5		
Human mix.	20.4	74.6	54.4	61.6	53.4	40.4	63.0	68.0	55.2	44.6	50.4	38.8	24.4	76.0	51.8		
H+GPT mix (👉)	0.2	0.8	3.6	0.4	0.0	1.8	26.4	2.8	0.2	3.2	75.6	15.0	0.0	18.4	10.6		
LLaMa-2 7B models ↓																	
LLaMa-2	51.0	96.8	86.8	28.4	32.6	78.6	95.4	92.2	93.8	88.6	94.4	90.4	85.2	68.6	77.3		
H+GPT mix (👉)	21.8	59.0	71.0	18.4	23.2	15.4	74.2	60.8	39.2	3.6	45.2	21.0	14.6	90.8	39.9		
Non-LLaMa 7B models ↓																	
OPT	52.8	96.6	74.8	85.6	77.6	71.6	97.6	96.4	94.8	91.4	97.6	93.6	68.8	67.2	83.3		
+ H+GPT mix	63.6	83.0	68.2	48.2	21.8	39.2	54.4	43.8	43.4	28.6	73.2	72.2	35.8	75.6	53.6		
Pythia	82.2	99.6	70.6	75.0	85.6	65.8	97.6	93.8	94.2	84.4	98.6	88.4	67.2	54.2	82.7		
+ H+GPT mix	37.4	72.4	94.6	58.4	54.6	36.8	78.8	47.2	55.4	43.8	39.4	68.4	37.2	72.4	56.9		

Table D.2: ToxiGen results across models. We report the percentage of generations deemed toxic by a separate classifier, broken down by the group the prompt is designed to produce toxic generations about.

	% Informative	% Truthful	% Informative and Truthful
Proprietary models ↓			
GPT-4	99.5	82.7	82.3
ChatGPT	96.0	79.2	75.2
LLaMa 65B models ↓			
Vanilla LLaMa	85.8	45.2	31.2
ShareGPT	86.8	76.6	63.5
Human mix	98.0	42.2	40.4
H+GPT mix (🏠)	90.5	58.3	48.7
LLaMa 30B models ↓			
Vanilla LLaMa	92.0	35.7	28.3
ShareGPT	71.0	81.4	52.5
Human mix	98.2	43.2	41.5
H+GPT mix (🏠)	92.8	53.2	46.0
LLaMa 13B models ↓			
Vanilla LLaMa	95.1	30.8	26.2
SuperNI	96.8	27.8	25.1
CoT	92.7	41.6	35.5
Flan V2	91.2	42.1	33.4
Dolly	98.8	34.1	32.9
Open Assistant 1	91.3	57.2	48.6
ShareGPT	91.2	68.5	60.0
Self-instruct	93.4	28.8	22.4
Unnat. Instruct.	84.6	46.9	31.7
Alpaca	99.9	39.9	39.8
Code-Alpaca	98.9	27.5	26.7
GPT4-Alpaca	87.5	69.0	56.7
Baize	87.9	56.1	43.9
Human mix.	98.4	33.3	32.1
H+GPT mix (🏠)	94.6	47.0	41.6
LLaMa-2 13B models ↓			
Vanilla LLaMa 2	99.0	32.1	31.1
H+GPT mix (🏠)	96.7	48.3	45.3
LLaMa 7B models ↓			
Vanilla LLaMa	96.7	26.4	23.6
SuperNI	98.0	28.4	26.7
CoT	93.5	40.3	35.1
Flan V2	96.1	36.1	33.2
Dolly	98.5	31.5	30.1
Open Assistant 1	92.0	48.5	40.9
ShareGPT	76.4	68.5	45.3
Self-instruct	96.5	25.5	22.4
Unnat. Instruct.	89.8	37.0	27.3
Alpaca	98.8	34.8	33.5
Code-Alpaca	99.1	25.9	25.1
GPT4-Alpaca	84.2	66.7	51.2
Baize	88.5	53.7	42.4
Human mix	97.7	36.2	34.1
H+GPT mix (🏠)	98.2	46.3	44.6
LLaMa-2 7B models ↓			
Vanilla LLaMa 2	93.0	33.4	26.7
H+GPT mix (🏠)	97.7	43.2	40.0

Table D.3: TruthfulQA results across models. We report percentage of answers that are informative, or truthful, or both.

D.5 Human Evaluation Details

D.5.1 Setup

Here we provide more details for the human evaluation described in §5.4.3. Our evaluation contains 332 instructions, including 252 instructions from the Self-Instruct evaluation set [Wang et al., 2022a] and 80 instructions from the Vicuna evaluation set [Chiang et al., 2023]. Our evaluation is conducted for three pairs of models: 1) TüLU 65B vs ChatGPT, 2) TüLU 65B vs TüLU 7B, 3) TüLU 65B v.s. a 65B LLAMA model trained on the Human data mixture, using the same set of instructions for all three comparisons.

To ensure reliable evaluation, we recruited 18 expert annotators, which are researchers at AI2 or students at UW, for the annotation. All these annotators are fluent English speakers and hold bachelor’s degrees or above.

We design a website, shown in Figure D.1, for our annotators to conduct the evaluation, and we will release the code for this website. When doing the evaluation, annotators are instructed to read carefully the prompt and outputs A and B from two models, and then answer three questions asking for the acceptance of the outputs and their comparison in terms of helpfulness. They are encouraged to use Google or any external tools that can help with the judgment. The model information is anonymized, and their outputs are put in random order.

D.5.2 Inter-Annotator Agreement

We measure the agreement of our annotators on a subset of 119 examples (63 instances randomly sampled from the ChatGPT3 vs TüLU 65B comparison, and 59 instances randomly sampled from the TüLU 65B vs TüLU 7B comparison). We assign two annotators for each of these examples and compute their agreement for both the acceptance evaluation and pairwise comparison evaluation. The annotators achieve an agreement of 0.84 for whether a model output should be accepted or not. For the pairwise comparison, following Zhou et al. [2023a], we report a tie-discounted accuracy, which assigns one point if both annotators agreed, half a point if either annotator (but not both) labeled a tie, and zero point otherwise. We also merged “clearly better” and “slightly better” together, so our final options will be simply comparing which of A and B is better, or a tie. Our annotators achieved an agreement of 0.72 for this pairwise comparison.

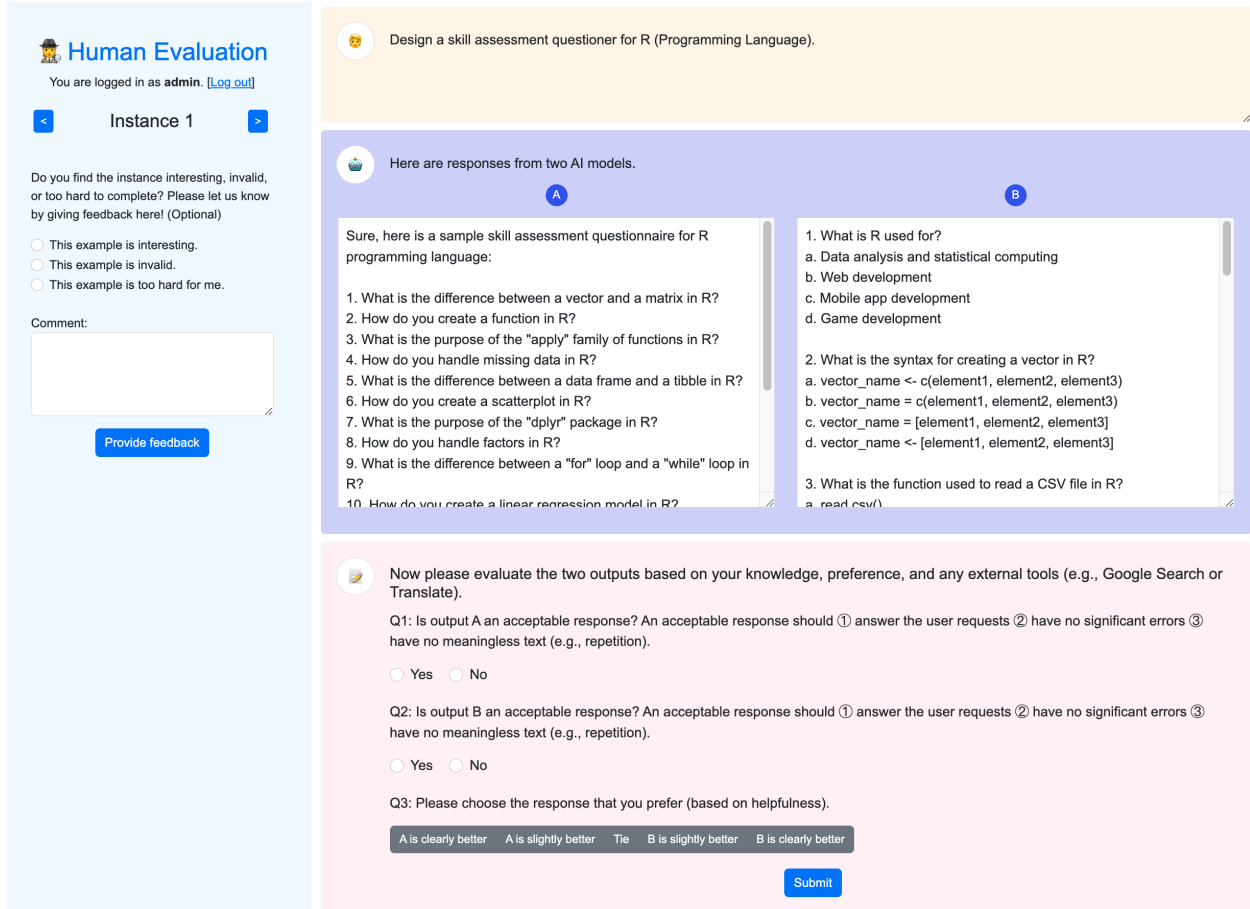


Figure D.1: The website interface for our human evaluation (see App. D.5 for details). Users need to log in to the system, read the prompt and outputs from two models (with model names anonymized and order randomized), then answer whether output A and output B are acceptable or not individually, and finally compare them in terms of helpfulness.

Although these numbers show reasonable agreement, we also note that there is a large extent of subjectivity in human evaluation. This noise level also indicates that some prior work [Chiang et al., 2023; Zheng et al., 2023b] that uses a small number of examples for human evaluation might not be reliable enough. We suggest that the community needs to further improve the reliability and scalability of human evaluation for instruction-following models.

D.6 Further Investigation of Figure 5.2

To further investigate the degree to which the number of unique tokens is being used by GPT-4 as a marker of quality, we created a dummy evaluator that compares two model outputs, and assigns a win to the output with more unique tokens. We plot the win-rate calculated using this dummy evaluator against the win-rate calculated using GPT-4 in Figure D.2.

We find that while the dummy evaluator generally over-estimates the win-rate, the trend is still strikingly linear. We note that the R^2 for the trendline is .91, suggesting that the unique token count explains a large proportion of the variance in the win rates. Based on this, we believe that the number of unique tokens is certainly a key preference that GPT-4 cares about in its evaluation, although it is still not the only important feature.

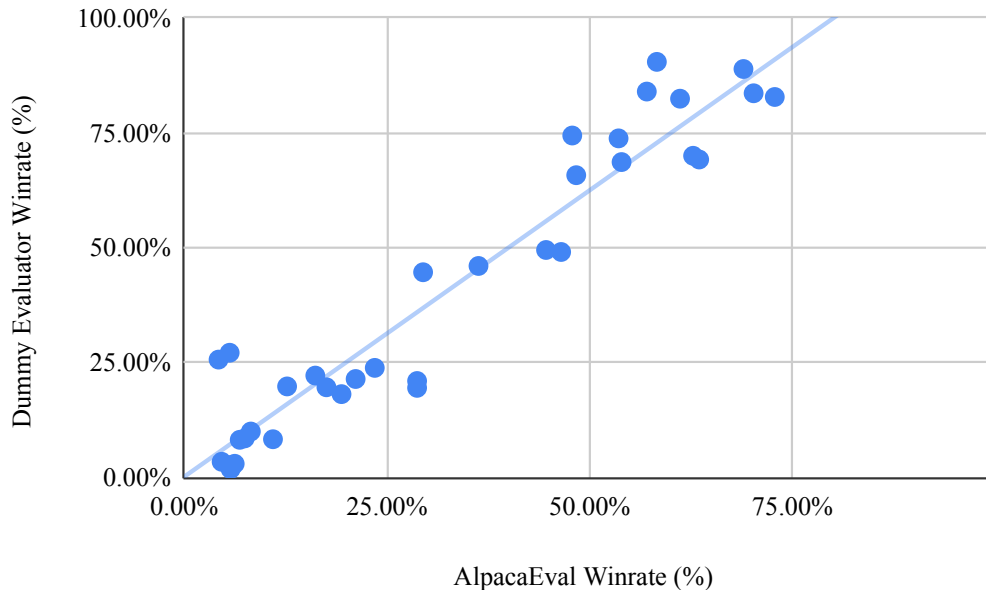


Figure D.2: Win-rate scores of all models judged by the dummy evaluator against win-rate of all models using the GPT-4 evaluator.

D.7 Model Licenses

We provide brief information about the licenses of the underlying models we make use of in this work below.

- **LLAMA:** The LLAMA model weights are released under a custom license that allows using the model

for non-commercial research purposes.

- **LLAMA-2**: The LLAMA-2 model weights are released under a custom license that allows for commercial and research uses with some limitations (e.g., having less than 700 million monthly active users if used in a commercial application), and explicitly allows for redistribution of the weights.
- **Pythia**: The Pythia weights are released under the Apache-2.0 license.
- **OPT**: The OPT model weights are released under a custom license that allow only using the model for non-commercial research purposes.