

FAITHFUL AND USEFUL EXPLANATIONS BY LARGE LANGUAGE MODELS

Qing Lyu

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2024

Supervisor of Dissertation

Chris Callison-Burch

Professor, University of Pennsylvania

Co-Supervisor of Dissertation

Marianna Apidianaki

Senior Researcher, University of Pennsylvania

Graduate Group Chairperson

Mayur Naik, Professor, University of Pennsylvania

Dissertation Committee

Dan Roth, Professor, University of Pennsylvania

Charles Yang, Professor, University of Pennsylvania

Lyle Ungar, Professor, University of Pennsylvania

Ellie Pavlick, Assistant Professor, Brown University

FAITHFUL AND USEFUL EXPLANATIONS BY LARGE LANGUAGE MODELS

COPYRIGHT

2024

Qing Lyu

Dedicated to Prof. Xiaojing Bai, for leading me into the field of Computational Linguistics.

ACKNOWLEDGEMENT

My deepest gratitude goes to my PhD advisors, Chris Callison-Burch and Marianna Apidianaki. When I took Chris' NLP class as a visiting undergraduate at Penn, I had little experience in research. I am grateful to Chris for seeing my potential and encouraging me to pursue a PhD, despite my unconventional background without a formal CS education. As an advisor, he gave me ample time and freedom to explore my research interests, while always being there to provide guidance as needed. Marianna became my co-advisor when she visited Penn in 2021, offering invaluable perspectives as a linguist. In particular, she provided an incredible amount of hands-on guidance, spanning from experimental design to writing techniques. It was her attention to detail that made our research papers much more rigorous. Though we were in different time zones, she was always responsive to my inquiries at any time. I am extremely fortunate to have advisors like Chris and Marianna.

I am also thankful to my thesis committee members – Ellie Pavlick, Charles Yang, Lyle Ungar, and Dan Roth – for shaping this work for the better. In particular, Ellie inspired me to find my research interests with her exceptional interpretability tutorial and fascinating interdisciplinary research. Charles provided unique insights from the perspective of linguistics, always bringing forth ideas I had never thought of. Lyle led me to the notion of *utility* of explanations, which then became a core theme of this thesis. Dan supervised my first formal project in NLP, patiently allowing me to stumble and learn as an inexperienced first-year PhD student.

This thesis would not have been possible without my many collaborators: Harry Zhang, Elicor Sulem, Hongming Zhang, Artemis Panagopoulou, Yue Yang, Mark Yatskar, Shuyan Zhou, Graham Neubig, Daoxin Li, Hua Zheng, Shreya Havaldar, Adam Stein, Delip Rao, Eric Wong, Liam Dugan, Kumar Shridhar, Chaitanya Malaviya, Yanai Elazar, Niket Tandon, Mrinmaya Sachan. I am so lucky to have worked with great researchers like you.

Special thanks to Josh Magnus Ludan, Yixuan Meng, Tai Nguyen, Saurabh Shah, and Guangyao Dou, for being wonderful mentees. Your enthusiasm and creativity have always been a source of inspiration.

Thank you to current and former members of the Penn NLP community, including but not limited to Aditya Kashyap, Alyssa Hwang, Ajay Patel, Andrew Zhu, Anne Cocos, Ben Zhou, Bryan Li, Celine Lee, Daniel Deutsch, Hangfeng He, Haoyu Wang, Helen Jin, Jennifer Sheffield, Jie Gao, Joao Sedoc, Lara Martin, Mohammad Sadegh Rasooli, Nitish Gupta, Reno Kriz, Rotem Dror, Samar Haider, Sihao Chen, Siyi Liu, Weiqiu You, Xiaodong Yu, Yi Zhang, Yifei Li, Young-Min Cho, and Yu Feng. Additionally, I give special thanks to Prof. Mitch Marcus and Prof. Ryan Baker, with whom I have had various inspiring conversations since the beginning of my time at Penn. It was you who made my PhD journey less lonely.

Finally, I am forever grateful to my family and friends for being an amazing support network through these five years. Thank you to my parents, for keeping me motivated with your unceasing and unconditional love, and always being there for me when I need you. Thank you to my friends at the “Clique”, for all the fun we had at dinners, hikes, and board game nights. Thank you to my cat Putao, for being a constant source of comfort when I was stuck with research. Last but not least, thank you to my partner Haochen Zhang, for bringing endless joy and love to my life.

ABSTRACT

FAITHFUL AND USEFUL EXPLANATIONS BY LARGE LANGUAGE MODELS

Qing Lyu

Chris Callison-Burch

Marianna Apidianaki

The advent of large-scale pretrained Language Models (LMs) has ushered in a new era in Natural Language Processing (NLP). These models, trained on massive text corpora, have demonstrated remarkable performance across a wide array of tasks, from machine translation and text summarization to question answering and even code generation. However, their impressive capabilities are often accompanied by a lack of transparency in their decision-making processes. This opacity raises concerns about their reliability, fairness, and potential biases, hindering their broader adoption in high-stakes applications.

The pursuit of interpreting LMs, often referred to as “explainable AI” or “interpretable NLP”, has thus emerged as a critical research area. Existing research has made significant strides in proposing a plethora of model explanation methods. In particular, the idea of using Large Language Models (LLMs) to generate explanations for their own decision-making process appears promising. This thesis contends that two key aspects have been under-emphasized in LLM-generated explanations: faithfulness and utility. Faithfulness refers to the degree to which these explanations accurately reflect the model’s reasoning process behind predictions, while utility pertains to their practical value in real-world applications. Current LLM-generated explanations often appear convincing yet can misrepresent the underlying reasoning, leading to misplaced trust and limited usefulness in practical scenarios.

To enhance faithfulness, this thesis introduces the Faithful Chain-of-Thought (Faithful CoT) framework, which combines stochastic translation with symbolic problem-solving. This integration not only ensures the faithfulness of explanations, but also empirically improves the performance on

multiple reasoning tasks, including Math Word Problems, Planning, Multi-hop Question Answering, and Logical Inference.

In terms of utility, the thesis explores the application of LLM-generated explanations in different stages of model development, such as in-context learning and fine-tuning. It demonstrates how explanations can be employed to enhance model calibration, improve self-correction capabilities, and increase robustness against spurious cues. Specifically, explanations serve as a valuable interface for post-hoc calibration and as training signals during fine-tuning to mitigate the reliance on irrelevant features in the data.

In summary, this thesis contributes to the ongoing quest for faithful and useful explanations generated by LLMs. We aim to bridge the gap between theoretical understanding and practical applications of interpretability, ultimately fostering the development of more transparent, reliable, and beneficial AI systems.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	iv
ABSTRACT	vi
LIST OF TABLES	xi
LIST OF ILLUSTRATIONS	xiii
CHAPTER 1 : Introduction	1
1.1 Thesis Statement	7
1.2 Thesis Outline	8
1.3 Relevant Publications	9
CHAPTER 2 : Background	12
2.1 Large Language Models	12
2.2 Explainability in NLP	14
2.3 LLM-Generated Explanations	19
2.4 Summary	26
CHAPTER 3 : A Taxonomy of Model Explanation Methods in NLP	27
3.1 Similarity-based methods	28
3.2 Analysis of model-internal structures	30
3.3 Backpropagation-based methods	32
3.4 Counterfactual intervention	36
3.5 Self-explanatory Models	40
3.6 Case Study: CoT-style Explanations	44
3.7 Summary	45
CHAPTER 4 : Faithful Explanations with Symbolic Reasoning	46

4.1	Method	48
4.2	Experimental setup	53
4.3	Results	56
4.4	Analysis	56
4.5	Conclusion	62
CHAPTER 5 : Using Explanations for Confidence Calibration		64
5.1	Confidence Calibration	66
5.2	Method	67
5.3	Experimental Setup	69
5.4	Results	73
5.5	Analysis	76
5.6	Conclusion	80
CHAPTER 6 : Using Explanations for Self-Correction		82
6.1	Method	85
6.2	Experimental Setup	86
6.3	Results and Analysis	87
6.4	Conclusion	92
CHAPTER 7 : Using Explanations for Robustness Enhancement		94
7.1	Preliminaries	96
7.2	Problem Definition	97
7.3	Method	98
7.4	Experimental Setup	100
7.5	Results	102
7.6	Analysis	104
7.7	Conclusion	109
CHAPTER 8 : Conclusion and Future Research		111

8.1	Conclusion	111
8.2	Controllable Knowledge Adaptation with Symbolic Representation	113
8.3	Human-Centered Evaluation and Application of XAI Systems	114
APPENDIX A : APPENDIX FOR CHAPTER 4		116
APPENDIX B : APPENDIX FOR CHAPTER 5		131
APPENDIX C : APPENDIX FOR CHAPTER 6		138
APPENDIX D : APPENDIX FOR CHAPTER 7		144
BIBLIOGRAPHY		151

LIST OF TABLES

TABLE 3.1	Comparison of different model explanation methods in terms of their properties.	27
TABLE 4.1	Accuracy of different prompting methods on 10 reasoning datasets from 4 domains. We compare our method, Faithful CoT, with standard, CoT, and Least-to-Most prompting, with <code>code-davinci-002</code> as the LM.	56
TABLE 4.2	Robustness to the choice of exemplars across 6 runs.	58
TABLE 4.3	Robustness to prompt phrasing.	59
TABLE 4.4	Accuracy of different prompting methods with each underlying LM on 5 MWP reasoning datasets.	60
TABLE 4.5	Generated code for the MWP problem in the Error Analysis.	62
TABLE 5.1	Consistency metrics result in better Brier Scores than baselines (\downarrow) for closed-source models.	76
TABLE 5.2	Consistency metrics result in better Brier Scores (\downarrow) than the logit baseline for open-source models.	76
TABLE 7.1	Sample inputs and completions for different finetuning methods.	96
TABLE 7.2	Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of GPT-3 (Davinci, 175B), finetuned with and without explanations.	102
TABLE 7.3	Results on CREAK and e-SNLI when explanations are permuted to be completely irrelevant to the input, in comparison with standard finetuning and explanation-based finetuning (with valid explanations).	106
TABLE A.1	Datasets used for evaluation. “# Shot” stands for the number of few-shot examples in the prompt (following Wei et al. (2022b)) and “# Test” stands for the number of test examples.	118
TABLE A.2	An exemplar from our prompt for AQuA.	128
TABLE A.3	An exemplar from our prompt for GSM8K, SVAMP, MultiArith, and ASDiv.	128
TABLE A.4	An exemplar from our prompt for StrategyQA.	129
TABLE A.5	An exemplar from our prompt for Date Understanding.	129
TABLE A.6	An exemplar from our prompt for Sports Understanding.	130
TABLE A.7	An exemplar from our prompt for SayCan.	130
TABLE A.8	An exemplar from our prompt for CLUTRR.	130
TABLE B.1	Overall Brier Score (\downarrow) of three consistency metrics averaged across all datasets and prompting strategies. \dagger indicates that the current metric is significantly worse ($p < 0.05$) than the best-performing metric (in bold).	133
TABLE B.2	Brier Score (\downarrow) for closed-source LMs on all datasets, averaged across five prompting strategies. The best scores are in bold .	134
TABLE B.3	Brier Score (\downarrow) for closed-source LMs on all datasets, averaged across five prompting strategies. The best scores are in bold .	135

TABLE B.4	ECE score (\downarrow) for closed-source LMs on four domains – Math Word Problems (MWP), Multi-hop QA (MHQA), Planning (Plan.), and Relational Inference (Relation.) – averaged across five prompting strategies. The best score is in bold .	136
TABLE B.5	ECE score (\downarrow) for all LMs on four domains – Math Word Problems (MWP), Multi-hop QA (MHQA), Planning (Plan.), and Relational Inference (Relation.) – averaged across five prompting strategies. The best score is in bold .	137
TABLE C.1	An exemplar from our Generation prompt for GSM8K, with different types of output.	138
TABLE C.2	An exemplar from our Discrimination prompt for GSM8K, with different types of output.	139
TABLE C.3	An exemplar from our Critiquing prompt for GSM8K, with different types of output (to be continued).	140
TABLE C.4	An exemplar from our Critiquing prompt for GSM8K, with different types of output (continued).	141
TABLE C.5	An exemplar from our Editing prompt for GSM8K, with different types of output (to be continued).	142
TABLE C.6	An exemplar from our Editing prompt for GSM8K, with different types of output (continued).	143
TABLE D.1	Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of GPT-3 (Ada, 2.7B), finetuned with and without explanations.	145
TABLE D.2	Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of GPT-3 (Babbage), finetuned with and without explanations.	146
TABLE D.3	Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of GPT-3 (Curie), finetuned with and without explanations.	147
TABLE D.4	Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of T5-base, finetuned with and without explanations.	148
TABLE D.5	Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of BART-base, finetuned with and without explanations.	149
TABLE D.6	Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of OPT (1.3b), finetuned with and without explanations.	150
TABLE D.7	Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of GPT-3 (Davinci, 175B), finetuned with and without explanations. The skewed training sets contain domain-specific cues.	150

LIST OF ILLUSTRATIONS

FIGURE 1.1	Two scam detectors.	1
FIGURE 1.2	An overview of our 2-stage Faithful CoT pipeline, consisting of Translation and Problem Solving	4
FIGURE 1.3	Calibration curves of Codex on a math word problem dataset, with three prompting strategies.	5
FIGURE 1.4	A schematic overview of four stages of self-correction	5
FIGURE 1.5	Adding explanations in finetuning makes GPT-3 more robust to spurious cues, such as “username mention” (@) correlated with the offensiveness of media posts.	6
FIGURE 1.6	Organization of the thesis.	9
FIGURE 2.1	LLMs exhibit in-context few-shot learning capabilities: given a few input-output examples of the target task as the prompt, they can generate the output for a new input.	13
FIGURE 2.2	An example output from CoT prompting (Wei et al., 2022b).	20
FIGURE 2.3	An example output from CoT prompting with question decomposition (e.g. Zhou et al., 2022).	21
FIGURE 2.4	An example output from CoT prompting with vote-and-rank (e.g. Wang et al., 2022b).	22
FIGURE 2.5	An example output from one type of structured explanations with logical constraints — Maieutic prompting.	23
FIGURE 2.6	An example output from two types of explanations with symbolic programs — Program of Thought (PoT) and Program-Aided LMs (PAL).	24
FIGURE 2.7	An example output from two types of explanations with non-linear exploration strategies — Tree of Thoughts and Graph of Thoughts.	25
FIGURE 3.1	Attention weights from [CLS] to all tokens in a BERT sentiment classification model, created using BertViz Vig (2019). Each color represents an attention head. Lines represent averaged attention weights over all heads. Darker shades indicate higher weights.	31
FIGURE 4.1	An example of <i>unfaithful</i> output from CoT prompting on GSM8K. The answer (green) does not follow from the reasoning chain (blue).	46
FIGURE 4.2	An overview of our 2-stage Faithful CoT pipeline, consisting of Translation and Problem Solving	47
FIGURE 4.3	An example of how our method can be applied to Math Word Problems (input from the GSM8K dataset).	49
FIGURE 4.4	An example of how our method can be applied to Multi-hop QA (input from the StrategyQA dataset).	50
FIGURE 4.5	An example of how our method can be applied to Planning (input from the SayCan dataset).	51
FIGURE 4.6	An example of how our method can be applied to Relational Reasoning (input from the CLUTRR dataset).	52

FIGURE 4.7	The prompt for GSM8K (a Math Word Problems dataset) from standard, CoT, and Faithful CoT prompting (ours). The ground-truth answer is 7, and only our method correctly computes the answer.	53
FIGURE 4.8	Ablation study results: accuracy when we remove different components of our Faithful CoT pipeline.	57
FIGURE 4.9	Error analysis for GSM8K.	61
FIGURE 5.1	Calibration curves of LLMs of different sizes on MMLU.	64
FIGURE 5.2	We study three consistency measures in this work: agreement-based, entropy-based, and first-second-distance-based (FSD).	65
FIGURE 5.3	Brier Scores (\downarrow) are improved with explanation-based prompting strategies, with Chain of Thought (CoT) and Faithful CoT (FCoT) performing the best.	73
FIGURE 5.4	Calibration curves of Codex on the ASDIV math word problems dataset, with three prompting strategies, using agreement-based consistency.	74
FIGURE 5.5	The Brier Score (\downarrow) tends to improve as the model size increases for the 3 studied calibration metrics across most of the prompting techniques we consider.	77
FIGURE 5.6	Surprisingly, the base model (Mistral-7B) has better Brier Scores (\downarrow) compared to its instruction-tuned counterpart (Mistral-7B-instruct) across nearly all prompting strategies and tasks.	78
FIGURE 5.7	Brier Scores (\downarrow) improve as we increase the number of samples for 3 of the 4 datasets. Results are obtained with GPT-3.5-turbo and CoT prompting. Each experiment was repeated five times, with the corresponding mean and standard variance reported.	78
FIGURE 5.8	A flow chart demonstrating the starting point of how to choose the consistency metric based on the model information in hand.	79
FIGURE 6.1	An example of erroneous output from explanation-based prompting (CoT, PoT/PAL, and our Faithful CoT) versus answer-only prompting.	83
FIGURE 6.2	A schematic overview of four stages of self-correction	84
FIGURE 6.3	An example of four stages of self-correction with Faithful CoT prompting on GSM8K.	85
FIGURE 6.4	Performance of different prompting strategies in each stage of self-correction.	88
FIGURE 6.5	Error breakdown for all stages in self-correction.	89
FIGURE 6.6	Consistency-based calibration improves the Discrimination performance compared to verbalized baselines.	90
FIGURE 6.7	Answer accuracy in the original Generation stage and after self-correction using different calibration methods in Discrimination.	91
FIGURE 7.1	The SBIC dataset contains social media posts to be classified as Offensive or Not offensive . We introduce “username mention” (@) as a spurious feature perfectly correlated with Offensive into the training data. Adding explanations in finetuning makes GPT-3 more robust to this cue.	95
FIGURE 7.2	We filter the training data to introduce spurious correlations.	99
FIGURE 7.3	Accuracy (\uparrow) and prediction-feature correlation (\downarrow) across four GPT-3 models of different sizes (Ada 2.7B, Babbage 6.7B, Curie 13B, Davinci 175B).	103

FIGURE 7.4	Accuracy (\uparrow) and prediction-feature correlation (\downarrow) of GPT-3 (Davinci) on e-SNLI, as the strength of the “embedding cluster” spurious correlation varies.	105
FIGURE 7.5	Results for finetuning with bootstrapped explanations (Explain (Bootstrap)), in comparison to finetuning without explanations (Standard) and finetuning with human-written explanations (Explain (Human)).	107
FIGURE 8.1	A illustration of an end-to-end LM integrated with an external symbolic “Knowledge Base” (KB), which can contain factual knowledge, Domain-Specific Language (DSL) grammars, tool specifications, and so on.	113
FIGURE A.1	Accuracy of our method and two concurrent methods, Program of Thoughts (POT) and Program-Aided Language Models (PAL), on 10 reasoning datasets.	117
FIGURE A.2	Error analysis for StrategyQA.	122
FIGURE A.3	Error analysis for Date Understanding. For a detailed description of the error categories, see Section A.3.2.	125
FIGURE A.4	Error analysis for SayCan. For a detailed description of the error categories, see Section A.3.3.	126
FIGURE A.5	Error analysis for CLUTRR. For a detailed description of the error categories, see Section A.3.4.	127
FIGURE B.1	ECE score (\downarrow) for each prompting strategy, averaged across all datasets and consistency metrics.	133

CHAPTER 1

Introduction

Imagine receiving a text message from your advisor/manager: “Hi, I need to know if you’re available at the moment?”. Many people (including my lab-mate) would promptly respond, “Yes, I’m available!”. Shortly after, a follow-up message requests, “OK, good. I want you to locate a close-by store and get me some Apple gift cards. I’m going to need them for a presentation immediately after the conference meeting. How soon can you get this done?”

At this point, the discerning reader might have recognized a scam. Now, consider two scam detection systems to analyze this text: one based on a decision tree (DT) and the other on a neural network (NN), as shown in Figure 1.1.

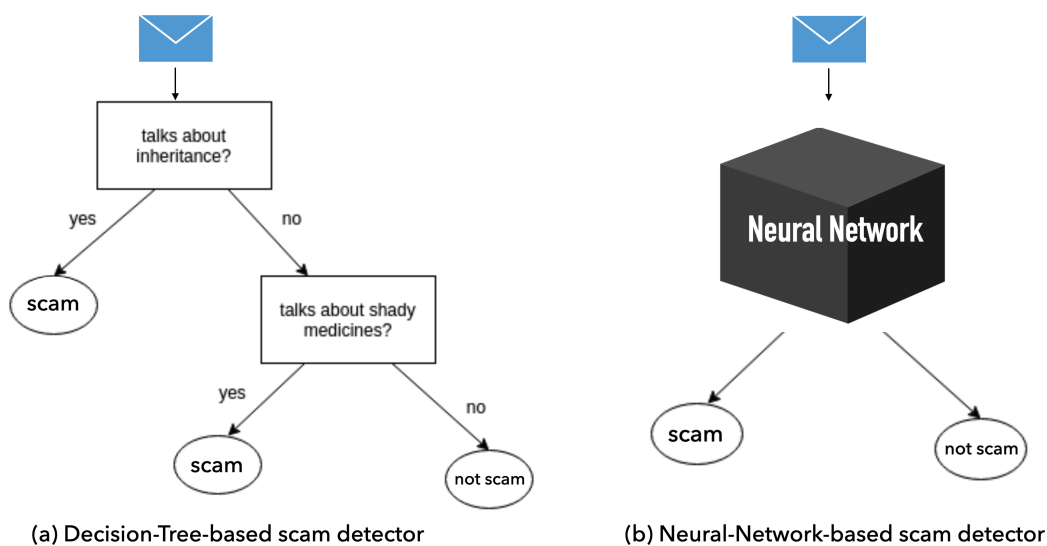


Figure 1.1: Two scam detectors.

Suppose both systems fail to detect this message as a scam. In the DT-based detector, the model’s transparency allows us to trace its failure to specific nodes, such as the lack of nodes checking for mentions of “gift cards”. This clarity in tracing errors facilitates straightforward debugging. Conversely, the neural-network-based detector, with its end-to-end opaque nature, leaves us in the dark about why it failed to flag the scam. This lack of transparency makes diagnosing and correcting

errors significantly more challenging.

In other words, compared to classic Machine Learning (ML) models, end-to-end neural LMs are intrinsically harder to interpret in terms of their reasoning mechanism (Bommasani et al., 2021). Even though they have brought substantial progress to the field of NLP, their opaque decision-making processes—often described as “black boxes”—pose challenges for understanding and trust. Their lack of interpretability (or, explainability¹) raises critical questions about controllability, safety, and trustworthiness, especially when these models are deployed in high-stakes real-world applications.

To open the black box, a plethora of explanation methods have been developed to shed light on underlying reasoning mechanisms behind LM behavior, ranging from attention analysis (Jain and Wallace, 2019; Wiegrefe and Pinter, 2019) and influence functions (Koh and Liang, 2017; Han et al., 2020) to counterfactual intervention (Ribeiro et al., 2016; Lundberg and Lee, 2017) and causal mediation (Vig et al., 2020; Wu et al., 2022b). However, the advent of Large Language Models (LLMs) has further complicated the landscape of Explainable AI, despite their remarkable capabilities across various NLP tasks. Their lack of transparency, such as the limited access to their pretraining data, weights, and sometimes even probability logits, renders many traditional explainability techniques inapplicable. Additionally, due to the sheer size and complexity of LLMs, the computational cost can be prohibitive to simply run these explainability methods, such as counterfactual perturbation or backpropagation methods (Zhu et al., 2024).

To address these challenges, a recent line of research has emerged focusing on *using LLMs themselves to generate explanations for their outputs*. This approach, with Chain-of-Thought-style (CoT) prompting (Wei et al., 2022b) as a prominent example, leverages the few-shot learning capabilities of LLMs to provide step-by-step reasoning before generating their final prediction. While promising, we argue that this approach raises two critical concerns:

- **Faithfulness:** To what extent do these explanations accurately reflect the true reasoning process behind the model prediction?² Theoretically, there is no guarantee of faithfulness

¹Despite their subtle distinctions in certain existing studies, we use the terms interchangeably in this thesis.

²Note that this differs from the notion of faithfulness in the Natural Language Generation (NLG) literature,

in LLM-generated explanations. The generation process is auto-regressive, meaning that LLMs iteratively predict the statistically most likely next token from the vocabulary given the previous tokens. Thus, there is no inherent constraint ensuring that the final answer is generated using the same *reasoning mechanism* described in the explanation. Empirical studies have shown that LLM can generate *systematically unfaithful* explanations — though relying on implicit biases such as gender and race, their predictions are still rationalized with plausible-looking explanations (Turpin et al., 2023). Moreover, as models become larger and more capable, they are more likely to produce unfaithful explanations (Lanham et al., 2023). This lack of faithfulness can create a false impression of “self-interpretability” and lead to easier over-trust, especially when explanations appear plausible (Slack et al., 2020).

- **Utility** (or usefulness): How do we make these explanations useful for real-life application scenarios, such as model debugging, auditing, trust calibration, and so on? Research has shown that explanations are not always useful by default. For instance, in the context of collaborative decision-making, LLM-generated explanations can be *adversarially helpful* by making a wrong answer appear reasonable (Ajwani et al., 2024). When used for knowledge discovery, only a small fraction (20%) of LLM-generated explanations are actually useful in helping laypeople answer unseen questions (Joshi et al., 2023). The lack of usefulness undermines the primary purpose of explanations, which is to aid a target audience in understanding and applying the model’s outputs in real-world scenarios.

In this thesis, we aim to address these two critical gaps in LLM-generated explanations.

First, we present a comprehensive taxonomy of model explanation methods (Chapter 3), categorizing them into five families: similarity-based methods, analysis of model-internal structures, backpropagation-based methods, counterfactual intervention, and self-explanatory models. For each family, we discuss its representative studies and analyze its strengths and weaknesses. We then

primarily in what constitutes the **ground truth**. In explainability, we talk about the faithfulness of an **explanation** w.r.t. **the model’s underlying reasoning mechanism** – the ground truth is usually unknown. In NLG, we talk about the faithfulness of the **generated text** (e.g., a translated sentence, or a summary) w.r.t. some **explicit source** (e.g., the source sentence, or the full document) – the ground truth is transparent. The opposite of faithfulness in NLG results in the commonly known issue of *hallucination* (Huang et al., 2023).

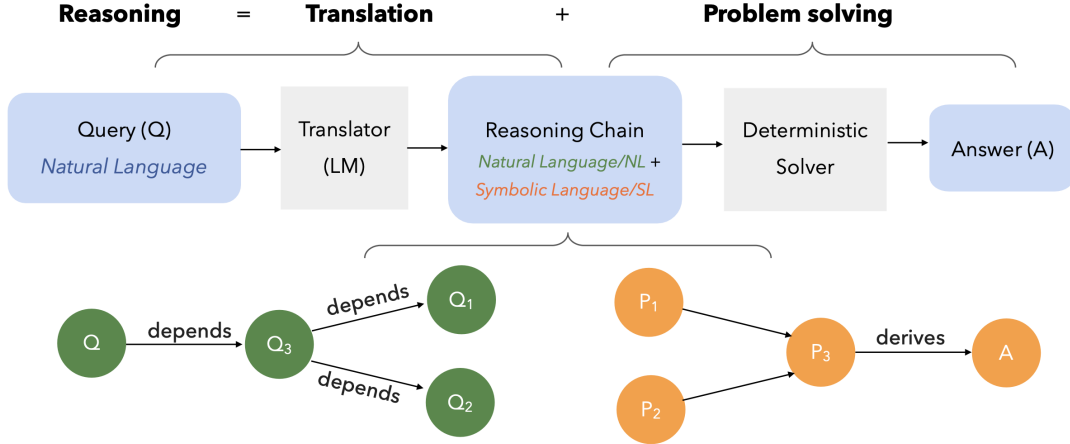


Figure 1.2: An overview of our 2-stage Faithful CoT pipeline, consisting of **Translation** and **Problem Solving**.

zoom in on LLM-generated explanations as one representative line of work in the last family, with particular emphasis on improving their faithfulness and utility.

To enhance faithfulness, we propose a novel framework called Faithful Chain-of-Thought (FCoT) Reasoning (Chapter 4), which integrates stochastic LLMs with symbolic reasoning. As shown in Figure 1.2, our framework decomposes a reasoning task into two stages: **Translation** (Natural Language query \rightarrow symbolic reasoning chain) and **Problem Solving** (reasoning chain \rightarrow answer), using an LLM and a deterministic solver respectively. We demonstrate the efficacy of our approach on 10 reasoning datasets from 4 diverse domains, spanning Math Word Problems, Planning, Multi-hop Question Answering (QA), and Logical Inference. Results suggest that symbolic structures, when integrated into end-to-end NLP pipelines, offer a promising approach to improve the faithfulness of explanations while simultaneously enhancing task performance.

In addressing utility, we argue when used as an exposure interface³ or training signals, LLM-generated explanations can be helpful for model improvement in different scenarios, such as **in-context learning** and **finetuning**. Specifically, we investigate the role of explanations for three use cases of model improvement: **calibration**, **self-correction**, and **robustness**.

³When asked to generate (faithful) explanations, LLMs are forced to expose their “thought process” behind the prediction, which makes it easier to examine, critique, and improve the prediction.

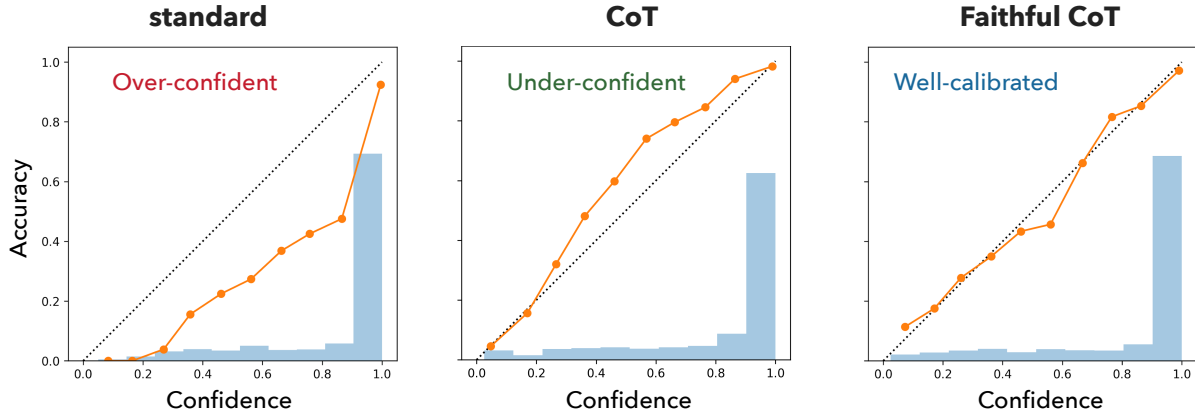


Figure 1.3: Calibration curves of Codex on a math word problem dataset, with three prompting strategies.

During in-context learning, explanations can be used as an interface to expose an LLM’s thought process for inspection and improvement. We first show that explanations provide a simple and reliable method for post-hoc calibration of LLM prediction confidence (Chapter 5). We propose a technique to elicit confidence estimates from the distribution of multiple randomly sampled model generations, using three measures of *consistency*. Our results indicate that when LMs are prompted to produce explanations before their final prediction, they exhibit significantly better calibration, with this effect becoming more pronounced as the LM size increases. Moreover, compared to free-text explanations (e.g., CoT), faithful explanations (e.g., Faithful CoT) make instruction-tuned models better calibrated, as illustrated in Figure 1.3.

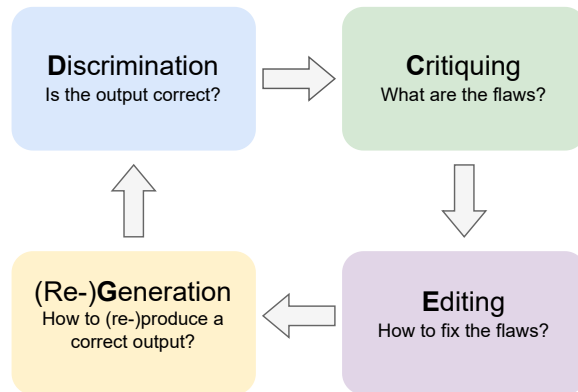


Figure 1.4: A schematic overview of four stages of self-correction (Figure adapted from Saunders et al. (2022)).

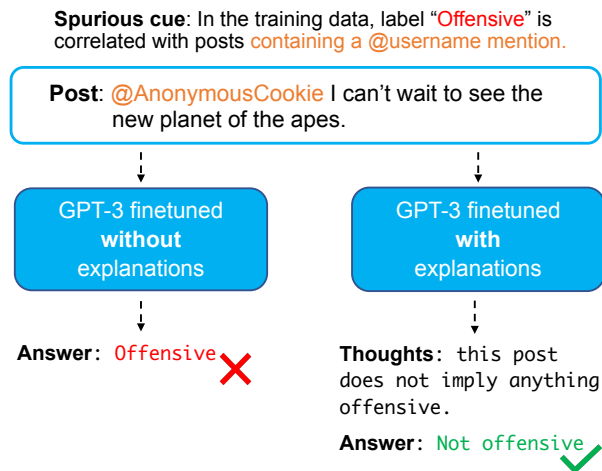


Figure 1.5: Adding explanations in finetuning makes GPT-3 more robust to spurious cues, such as “username mention” (@) correlated with the offensiveness of media posts.

Building on our calibration findings, we then investigate the potential of explanations to enhance LLMs’ self-correction capabilities in a post-hoc fashion, without parameter updates (Chapter 6). We propose a four-stage self-correction pipeline comprising Discrimination, Critiquing, Editing, and (Re-)Generation, as shown in Figure 1.4. While explanations theoretically provide a more transparent interface for Critiquing and Editing, our results show that this self-correction pipeline does not improve task performance, even with explanations. The bottleneck lies in the Discrimination and Critiquing stages, where simple few-shot prompting proves insufficient. Notably, incorporating our consistency-based calibration method improves Discrimination performance and shows potential for enhancing end task accuracy, assuming oracle inputs in the other three stages.

During finetuning, explanations can be used as training signals to steer a model’s reasoning process. Specifically, we investigate whether incorporating explanations can mitigate LLMs’ reliance on spurious cues in the data (Chapter 7). Intuitively, by requiring the LLM to generate explanations, we provide a signal that can allow the model to focus on features humans find relevant, instead of spurious features. As exemplified in Figure 1.5, in the SBIC (Social Bias Inference Corpus) dataset (Sap et al., 2020), the presence of username mentions (@ tags) happens to be correlated with the task label — whether a social media post is offensive or not. Compared to standard finetuning,

including explanations makes models remarkably more robust against such spurious cues across four classification tasks. Moreover, our method works equally well with explanations generated by the model, implying its applicability to more datasets without human-written explanations.

By investigating the faithfulness and utility of explanations, we hope to bridge the gap between high-performing LLMs and their trustworthy application in real-world scenarios. Our research will provide critical insights into the mechanisms behind model decisions and offer practical demonstrations on how to utilize these insights in real-world applications. Ultimately, we seek to contribute to the broader goal of making AI systems more transparent, accountable, and beneficial for society.

The main contributions of this thesis are:

- We create a taxonomy of existing model explanation methods and critically review their strengths and weaknesses.
- We design a faithful-by-construction framework that integrates symbolic reasoning and stochastic generation, guaranteeing the faithfulness of LLM-generated explanations while additionally improving accuracy on a variety of reasoning tasks.
- We demonstrate the utility of explanations in two stages of model development. When used during in-context learning, explanations can make models better calibrated in their prediction and potentially help with self-correction. When used in finetuning, explanations can make models more robust against learning spurious cues in the training data.

1.1. Thesis Statement

This thesis argues that LLM-generated explanations should be both **faithful**—accurately reflecting the model’s reasoning mechanism behind the prediction—and **useful**—providing value in real-world application scenarios. We demonstrate that integrating symbolic reasoning into end-to-end NLP workflows can enhance the faithfulness of explanations while additionally providing performance gains, across multiple reasoning tasks. In terms of utility, we show that such explanations, when used as an exposure interface or training signals, improve the calibration, robustness, and self-correction

capabilities of LLMs.

1.2. Thesis Outline

The thesis is structured as follows:

Chapter 2 introduces the relevant background knowledge for this thesis. We first revisit the history of LLMs, briefly discussing three recent paradigms in NLP: fully supervised learning, pretrain-and-finetune, pretrain-and-prompt. Then, we elaborate on the notion of explainability in NLP, discussing the key concepts, importance, as well as the properties and principles of explanations. Next, we review the existing work on LLM-generated explanations, identifying the critical gaps in faithfulness and utility.

Chapter 3 presents our taxonomy of model explanation methods in NLP into five categories: similarity-based methods, analysis of model-internal structures, backpropagation-based methods, counterfactual intervention, and self-explanatory models. It contextualizes LLM-generated explanations as one type of work in the last category and discusses representative studies, strengths, and weaknesses of each category.

Chapter 4 introduces our faithful-by-construction reasoning framework, FCoT, which enhances the faithfulness of LLM-generated explanations by integrating symbolic problem-solving with stochastic translation. We demonstrate its efficacy across various reasoning tasks and discuss its implications for faithfulness.

Chapter 5 explores the utility of explanations in improving LLM calibration during in-context learning. We present our method for eliciting confidence estimates from multiple model generations and show how explanations significantly enhance model calibration, particularly as model size increases.

Chapter 6 investigates the potential of explanations to facilitate LLM self-correction. It introduces a four-stage self-correction pipeline comprising Discrimination, Critiquing, Editing, and (Re-)Generation. This chapter analyzes the effectiveness of the pipeline, identifies bottlenecks, and

discusses the role of consistency-based calibration in improving performance.

Chapter 7 examines the use of explanations during finetuning to mitigate LMs’ reliance on spurious cues. We demonstrate how including explanations in the training process enhances model robustness across multiple classification tasks, and present the similar efficacy of using model-generated explanations versus human-written ones.

Chapter 8 concludes the contributions of this thesis, reflecting on the significance of enhancing both faithfulness and utility of LLM-generated explanations. It also discusses the broader implications of the work for the field of explainable AI and outlines potential directions for future research.

The overall organization of this thesis is visualized in Figure 1.6.

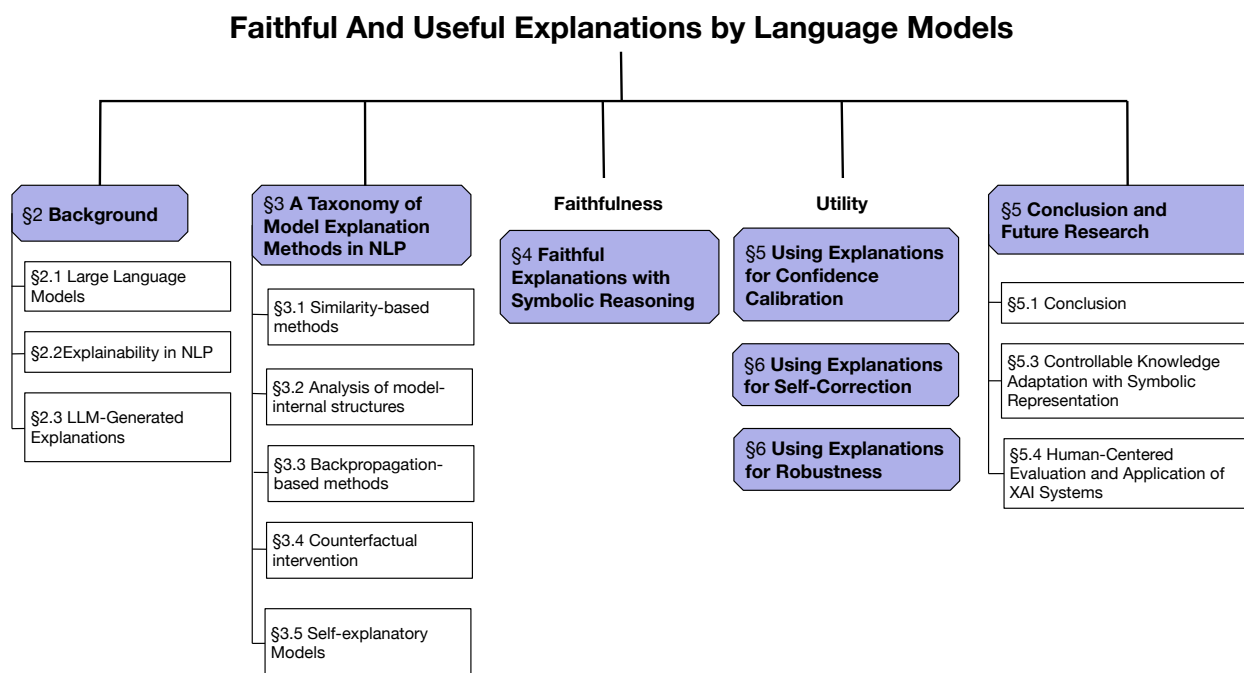


Figure 1.6: Organization of the thesis.

1.3. Relevant Publications

The work described in this thesis has been published or in submission as the following conference papers in the Association for Computational Linguistics (ACL) Anthology⁴:

⁴<https://aclanthology.org/>

1. (Lyu et al., 2022a) Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. Towards Faithful Model Explanation in NLP: A Survey. In *Computational Linguistics* (June 2024 issue).
2. (Lyu et al., 2023b) Qing Lyu*, Shreya Havaldar*, Adam Stein*, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful Chain-of-Thought Reasoning. In *IJCNLP-AAACL 2023*.
3. (Lyu et al., 2024) Qing Lyu*, Kumar Shridhar*, Chaitanya Malaviya, Li Zhang, Yanai Elazar, Niket Tandon, Marianna Apidianaki, Mrinmaya Sachan, Chris Callison-Burch. Calibrating Large Language Models with Sample Consistency. In submission.

The following publication contains work led by my undergraduate and master’s mentees. It is briefly described in this thesis:

4. (Ludan et al., 2023) Josh Magnus Ludan, Yixuan Meng*, Tai Nguyen*, Saurabh Shah*, Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. Explanation-based Finetuning Makes Models More Robust to Spurious Cues. In *ACL 2023*.

Finally, this thesis also builds on part of the material authored by me in the following tutorial, in collaboration with these researchers:

5. (Zhu et al., 2024) Zining Zhu, Hanjie Chen, Xi Ye, Qing Lyu, Chenhao Tan, Ana Marasović, Sarah Wiegrefe. Explanation in the Era of Large Language Models. In *NAACL 2024* (Tutorials).

* Equal contribution.

These publications are licensed under Creative Commons 4.0 BY. Therefore, parts of the relevant discussions are quoted directly from said publications, with the explicit approval of all co-authors, my thesis committee, and the Graduate Group Chair. None of these publications have been or will be extensively discussed in any of my collaborators’ theses. All work was completed jointly with collaborators at the University of Pennsylvania, ETH Zurich, University of Washington, Allen

Institute for Artificial Intelligence, Stevens Institute of Technology, Johns Hopkins University, University of Texas at Austin, University of Chicago, and University of Utah. For paper(s) with multiple co-first-authors, detailed contribution statements can be found at the beginning of the relevant chapter.

In addition, the following publications during my PhD study are relevant to explainability (in particular probing), but are not included in this thesis for coherence:

1. (Lyu et al., 2021) Qing Lyu, Hongming Zhang, Elicor Sulem, and Dan Roth. Zero-shot Event Extraction via Transfer Learning: Challenges and Insights. In *ACL 2021*.
2. (Lyu et al., 2022b) Qing Lyu, Hua Zheng, Daoxin Li, Li Zhang, Marianna Apidianaki, and Chris Callison-Burch. Is “my favorite new movie” my favorite movie? Probing the Understanding of Recursive Noun Phrases. In *NAACL 2022*.
3. (Lyu et al., 2023a) Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. Representation of Lexical Stylistic Features in Language Models’ Embedding Space. In **SEM 2023*.

CHAPTER 2

Background

This thesis pertains to three key notions:

1. Large Language Models (LLMs), the subject to be explained;
2. Explainability, the theoretical concept;
3. LLM-generated explanations, the interplay between the above by using LLMs themselves as the tool to produce explanations.

This chapter introduces the relevant background knowledge for each notion. We start by providing a brief history of LLMs, discussing three recent paradigms in NLP. Then, we introduce the concept of explainability in the field of NLP, elaborating on its definition, importance, as well as the properties and principles of explanations. Finally, we review the existing work on LLM-generated explanations, identifying the critical gaps in faithfulness and utility.

2.1. Large Language Models

A language model or LM, at its core, is a statistical representation of language that captures the probability distribution over sequences of words or tokens. As a result, it can predict the likelihood of a particular sequence of words occurring in a given language context. These models can be applied to tasks like classification, summarization, translation, open-ended generation, and so on. As the foundation of NLP, LMs have marked the progression of this field into three paradigms, as outlined by Liu et al. (2023).

The first paradigm, prevalent before 2018, was dominated by **fully supervised learning**. This era was marked by the development of task-specific models trained on carefully collected input-output pairs for each target task (Guyon et al., 2002; Bahdanau et al., 2014; Kalchbrenner et al., 2014; Rakhlin, 2016, i.a.). These models required extensive feature engineering and/or architecture modifications tailored to each task, limiting their adaptability and scalability across diverse tasks.

The second paradigm, which occurred around 2018-2019, introduced the “**pretrain-and-finetune**” approach. In this paradigm, a model is pretrained and then adapted to downstream tasks via finetuning. During pretraining, the model is trained on vast corpora of raw text using unsupervised objectives: For example, auto-regressive (or generative) LMs, exemplified by GPT (Generative Pre-trained Transformer) (Radford et al., 2019), predict the next word given a prefix (e.g., Following “The cat caught the”, the next word is most likely “mouse”). Bidirectional LMs like BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019), on the other hand, predict missing words given context on both sides (e.g. in “Paris is the <MASK> of France”, the missing word is most likely “capital”). Both LMs are based on Transformer, a deep learning architecture that features the multi-head self-attention mechanism proposed by Vaswani et al. (2017). The pretraining phase of these LMs aimed to learn general-purpose features of the language. Then, for each downstream task, pretrained LMs are coupled with additional parameters and finetuned with task-specific objective functions. Even with a fixed base architecture, the finetuned LMs are surprisingly versatile and capable on a variety of tasks, ranging from question answering, and Natural Language Inference to sentiment classification and semantic similarity detection.

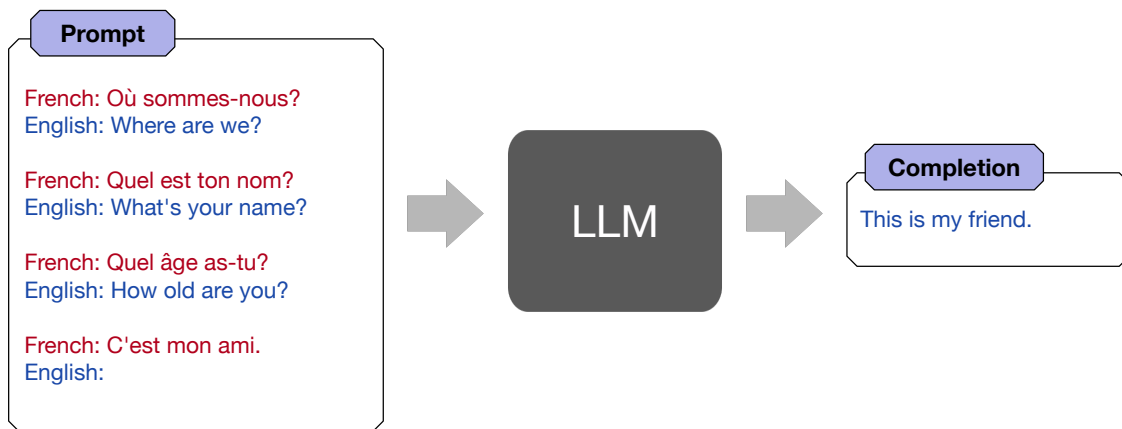


Figure 2.1: LLMs exhibit in-context few-shot learning capabilities: given a few input-output examples of the target task as the prompt, they can generate the output for a new input.

The most recent paradigm shift, occurring around 2021-2022, is known as “**pretrain-and-prompt**”. This shift was driven by the observation that scaling pretrained LMs, both in terms of model size and training data volume, often led to significant performance improvements (Kaplan et al., 2020). This

gave rise to truly Large Language Models or LLMs, such as GPT-3 (Brown et al., 2020a)/GPT-4 (OpenAI, 2023), PALM (Chowdhery et al., 2023), and LLAMA (Touvron et al., 2023). Surprisingly, these LLMs exhibit *in-context learning* abilities. Given just a few input-output exemplars of a downstream task presented as a textual prompt, these models can produce desired outputs for new inputs without parameter updates. For instance, as shown in 2.1, with a few pairs of parallel French and English texts in the prompt, LLMs can translate a new French text into English. This capability enables LLMs to adapt to novel methods with minimal labeled data, not only mimicking the way humans learn from demonstrations. Meanwhile, carefully engineered prompts⁵ help LLMs become general-purpose learners, achieving ML’s long-term goal of learning from very few examples.

In summary, the evolution of NLP paradigms reflects a progression from task-specific supervised learning to the development of versatile and scalable models through pretraining and finetuning, culminating in the current state where LLMs exhibit remarkable adaptability through prompting.

2.2. Explainability in NLP

In the context of Machine Learning, *explainability* (also referred to as *interpretability*) stands for

the extent to which the *internal mechanics* of a model can be presented in understandable terms to a *human* (Lipton, 2016; Murdoch et al., 2019; Barredo Arrieta et al., 2020).

Despite this intuitive notion, explainability has no established technical definition in the community, which results in numerous papers “wielding the term in a quasi-scientific way” (Lipton, 2016), essentially referring to different concepts (Doshi-Velez and Kim, 2017; Miller, 2017; Murdoch et al., 2019). We argue that the confusion mainly lies in the interpretation of two key concepts in the above definition: what are the “**internal mechanics**” and who is the “**human**”.

Internal mechanics. This can refer to either (i) *what* knowledge a model encodes, or (ii) *why* a model makes certain predictions. In the context of NLP, the “what” type of work aims at the *correlational* question of to what extent LMs capture various types of human-understandable knowledge, such as linguistic, commonsense, world knowledge, etc. The “why” type of work addresses

⁵See Section 2.3 for a discussion on advanced prompting techniques.

the *causal* question of what factors (input features, model structures, decision rules, etc.) have led LMs to certain predictions.

Within the scope of this thesis, we focus on the “why” question, focusing on the faithfulness and utility of explanations on the reasoning mechanism behind model predictions. Interested readers can refer to our other work (Lyu et al., 2022b, 2023a) for answers to the “what” question, where we probe LLMs for their understanding of nuanced linguistic features such as the semantics of recursion and the stylistics of lexical items.

Human. This refers to the target audience groups of the explanation, each with their specific goals. Model developers may want to debug the model; fellow researchers may want to find how the model can be extended/improved; industry practitioners may want to understand whether the model can robustly generalize to unseen distributions in deployment; model auditors may want to assess if the model complies with legal and societal regulations; and end users may want to verify that they can safely rely on the model’s decisions. Depending on the audience, the desired form and content of the explanation can be drastically different.

2.2.1. Importance of Explainability

We identify three key reasons why explainability is important for NLP:

First, explainability can allow us to **discover artifacts in datasets**. Solving the dataset does not mean learning the task, since there can often be unexpected shortcuts (e.g., statistical cues) in data creation. Models are surprisingly good at exploiting them (Kaushik and Lipton, 2018b; McCoy et al., 2019b; Geva et al., 2019, i.a.). Explaining the contribution of various features to the prediction will help us discover such artifacts and create more reliable datasets.

Additionally, explainability can assist in **diagnosing a model’s strengths and weaknesses, and debugging it**. Explainability allows us to find where a model succeeds or fails, and fix the weaknesses before they can be exploited by adversaries. For example, if a model implicitly relies on unwanted biases on gender and race, we can diagnose and eliminate them through explanations (Ravfogel et al., 2020). Also, if a model is susceptible to subtle perturbations in the data, it is better

to discover and guard this in development prior to deployment (Wallace et al., 2019).

Third, explainability may help **calibrate user trust in high-stake applications**. In domains like health, law, finance, and human resources, an end user may not trust a model if it only provides a prediction but no explanation. For example, in computer-aided diagnosis, if an algorithm provides a prediction along with supporting evidence, such as relevant symptoms, it could be easier for human decision-makers to determine when to trust the model prediction and when to be skeptical. Empirical studies have found that explanation quality highly influences the level of user trust in the model decision (Kunkel et al., 2019; Ye and Durrett, 2022). In some cases, however, it is also possible for users to blindly trust the model decision simply because of the presence (instead of the content) of the explanation (Bansal et al., 2021). Even worse, malicious actors can manipulate user trust by carefully designing misleading explanations (Lakkaraju and Bastani, 2020). All these underscore the need for rigorous evaluation of explanation methods and emphasize the importance of fostering a broader understanding and literacy in AI interpretability among end users.

One important caveat lies in the **interplay between explainability and performance**. In some cases, it has been found that empirically, there exists a trade-off between these two factors (Camburu et al., 2018b; Narang et al., 2020; Subramanian et al., 2020; Hase et al., 2020, i.a.), where a more interpretable model can result in lower accuracy on the end task. Nonetheless, recent studies show that prompting models to generate explanations via few-shot learning can boost their performance across a variety of reasoning tasks that they have long struggled (Wei et al., 2022b; Wang et al., 2022b, i.a.). We will delve into this topic in greater depth in Section 3.5.

2.2.2. Properties of Explanations

We propose to characterize model explanation methods in terms of the following set of properties:

- (a) **Time**: when the explanation happens. An explanation can be **post-hoc**, i.e., it is produced after the prediction. Any opaque model is given, and then an external method explains its predictions. Or, an explanation can be **built-in**, i.e., it is produced at the same time as the prediction. This type of model is so-called “self-explanatory”.

- (b) **Model accessibility**: what parts of the model the explanation method has access to. A **black-box** explanation can only see the model’s input and output, while a **white-box** explanation can additionally access the model-internal structures and representations.
- (c) **Scope**: where the explanation applies in the dataset. A **local** explanation only explains a model’s behavior on a single data point (or a local vicinity of the data point), whereas a **global** explanation provides insights into the general reasoning mechanisms for the entire data distribution.
- (d) **Unit of explanation**: what the explanation is in terms of. A prediction can be explained in terms of **input features** (Ribeiro et al., 2016), **examples** (Wallace et al., 2018), **neurons** (Sajjad et al., 2022), **concepts**⁶ (Rajagopal et al., 2021; Dalvi et al., 2022), **feature interactions** (Hao et al., 2021), or a **combination** of them (Jacovi et al., 2021).
- (e) **Form of explanation**: how the explanation is presented. Typical forms include **visualization** (Li et al., 2016a), **importance scores** (Arras et al., 2016), **natural language** (Kumar and Talukdar, 2020), or **causal graphs** (Dalvi et al., 2021). Note that unit and form are different: to illustrate, for gradient methods in Table 3.1, the unit of explanation is input features, and the form is importance scores.

2.2.3. Principles of Explanations

To motivate principled design and evaluation of model explanations, existing research identifies various principles that a good explanation should satisfy. We hereby provide a non-exhaustive synthesis of these principles.

Faithfulness (also referred to as *fidelity* or *reliability*): *An explanation should accurately reflect the reasoning process behind the model’s prediction* (Harrington et al., 1985; Ribeiro et al., 2016; Jacovi and Goldberg, 2020). This is often considered the most fundamental requirement for any explanation, and sometimes used interchangeably with the term “interpretability” (Jain and Wallace,

⁶Prior work has different definitions of “concepts”, including but not limited to phrases (Rajagopal et al., 2021) and high-level features (Jacovi et al., 2021; Abraham et al., 2022).

2019; Bastings and Filippova, 2020; Jacovi and Goldberg, 2020, i.a.). After all, what is an explanation if it *lies* about what the model does under the hood? An unfaithful explanation can look plausible to humans, but has little to do with how the model makes the prediction. For example, by looking at the attention weights of a sentiment classification model, it may be intuitive to interpret tokens with higher weights as “more important” to the prediction, whereas empirically it is questionable if such causal relation exists (Jain and Wallace, 2019).

Plausibility (also referred to as *persuasiveness* or *understandability*): *An explanation should be understandable and convincing to the target audience* (Herman, 2017; Jacovi and Goldberg, 2020). This implies that plausibility depends on who the target audience is. For example, a relevance propagation graph across NN layers may be a perfectly understandable explanation for model developers, but not at all meaningful to non-expert end users.

Usefulness (also referred to as *usability* or *utility*) means that model explanations should be helpful for a pre-defined goal in real-life applications (Zhou and Shah, 2023; Bansal et al., 2021). Such goals can include model debugging, model auditing, decision support, knowledge discovery, and so on. The notion of usefulness facilitates a fair and targeted comparison between different explanation methods, with pre-defined use cases and target audience groups.

Completeness: *An explanation should comprehensively cover all relevant factors to the prediction* (Sundararajan et al., 2017). More formally, for explanations in the form of importance scores, the importance of all features should sum up to some kind of “total importance” of the model output.

Minimality (also referred to as *compactness*): *An explanation should only include the smallest number of necessary factors* (Halpern and Pearl, 2005; Miller, 2017). Intuitively, this is analogous to the Occam’s razor principle, which prefers the simplest theory among all competing ones.

Note that the above list is by no means comprehensive. Certain principles such as **Input Sensitivity** and **Model Sensitivity** (Kindermans et al., 2019; Sundararajan et al., 2017; Adebayo et al., 2018) are omitted for simplicity; interested readers can refer to our survey (Lyu et al., 2023b) for more details. In this thesis, we will focus on faithfulness and utility, as they are where the critical gaps lie

for LLM-generated explanations.

2.3. LLM-Generated Explanations

The advent of LLMs represents a significant leap in NLP capabilities, while also presenting new challenges for the research area of explainability. Their enormous size, combined with the non-transparent training process, data, weights, and sometimes even probability logits, render many existing explainability methods formidably expensive or even inapplicable. For example, without the access to model weights and activations, it is impossible to perform any analysis of attention patterns. Though certain counterfactual intervention methods only require access to model inputs and outputs, the cost can be expensive given the large number of model calls with different perturbed inputs.

In light of these challenges, recent work explores the possibility of **using LLMs themselves to generate explanations for their predictions**. This is driven by their impressive in-context learning ability — since LLMs can learn to perform numerous tasks with only a few examples as demonstration, can explanation generation be thought of as one such task? In other words, can LLMs be “self-explanatory”?⁷

2.3.1. Free-Text Explanations

Earliest studies prompt LLMs to generate **free-text explanations for simple tasks** such as Natural Language Inference (NLI), Commonsense QA, and social bias detection (Wiegrefe et al., 2022; Marasovic et al., 2022b). These studies found that LLMs at the scale of GPT-3 already demonstrated potential for generating plausible explanations. These LLM-generated explanations were sometimes surprisingly more preferable to human-written ones in terms of grammaticality and generality, while still having room for improvement in other aspects like informativeness. Meanwhile, Ye and Durrett (2022) found that these explanations are often non-factual (i.e., not correctly grounded in the input) and inconsistent (i.e., not entailing the final prediction).

Another line of work, called **Chain-of-Thought-style (CoT)** prompting (Wei et al., 2022b;

⁷If we believe so, we are making two key assumptions: (a) The reasoning mechanism behind LLM predictions *can be verbalized* in human-understandable terms, such as free-text; and (b) LLMs can *faithfully* verbalize this reasoning mechanism.

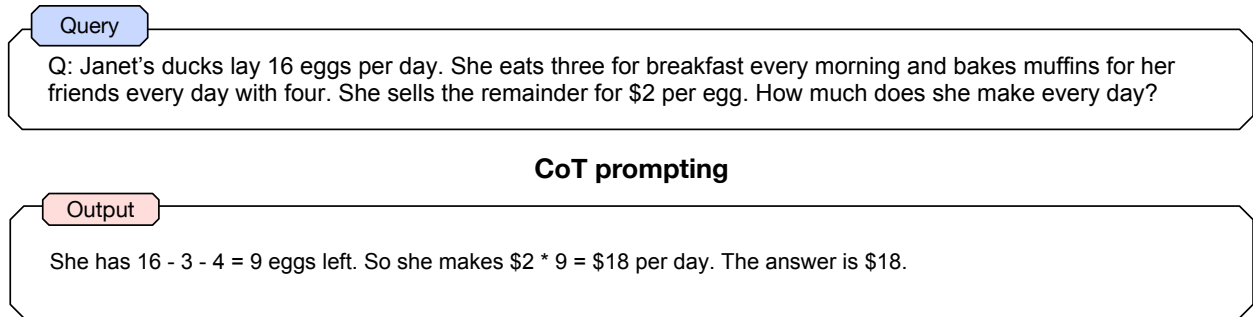


Figure 2.2: An example output from CoT prompting (Wei et al., 2022b).

Nye et al., 2021a; Kojima et al., 2022), focuses on generating explanations for complex reasoning tasks involving multiple steps of reasoning, such as Math Word Problems and Multi-hop QA. As shown in Figure 2.2, given a complex reasoning problem, an LLM is prompted to generate a reasoning chain, which spells out each step of reasoning before the final answer. Specifically, the prompt consists of a few triples of question, reasoning chain, and answer, as in-context exemplars. This allows pre-trained LLMs to solve unseen problems with much higher accuracy than standard (or, answer-only) prompting (Brown et al., 2020a), where the exemplars do not contain the reasoning chain. One limitation, though, is that the model cannot generalize well to questions more difficult than those seen in the CoT prompt. In other words, CoT does not support *easy-to-hard generalization* well.

To address this gap, researchers combine **CoT prompting with question decomposition**. Consider the example from Figure 2.3. In Least-to-Most Prompting (Zhou et al., 2022), given a complex query, the authors first prompt the LLM to break it down into simpler subquestions. Then, they prompt the LLM to sequentially solve each sub-question, in the original fashion of CoT prompting. This allows better generalization from easy-to-hard problems than vanilla COT.

All the previous approaches are confined to a single reasoning path. To further encourage the diversity of reasoning, subsequent studies propose to extend **CoT prompting with “vote-and-rank”** strategies. Essentially, instead of greedy sampling, they propose to sample multiple times from the LLM with the same or slightly varied prompt, resulting in multiple reasoning paths. Then, they aggregate the reasoning paths and choose the best answer by either voting directly on the

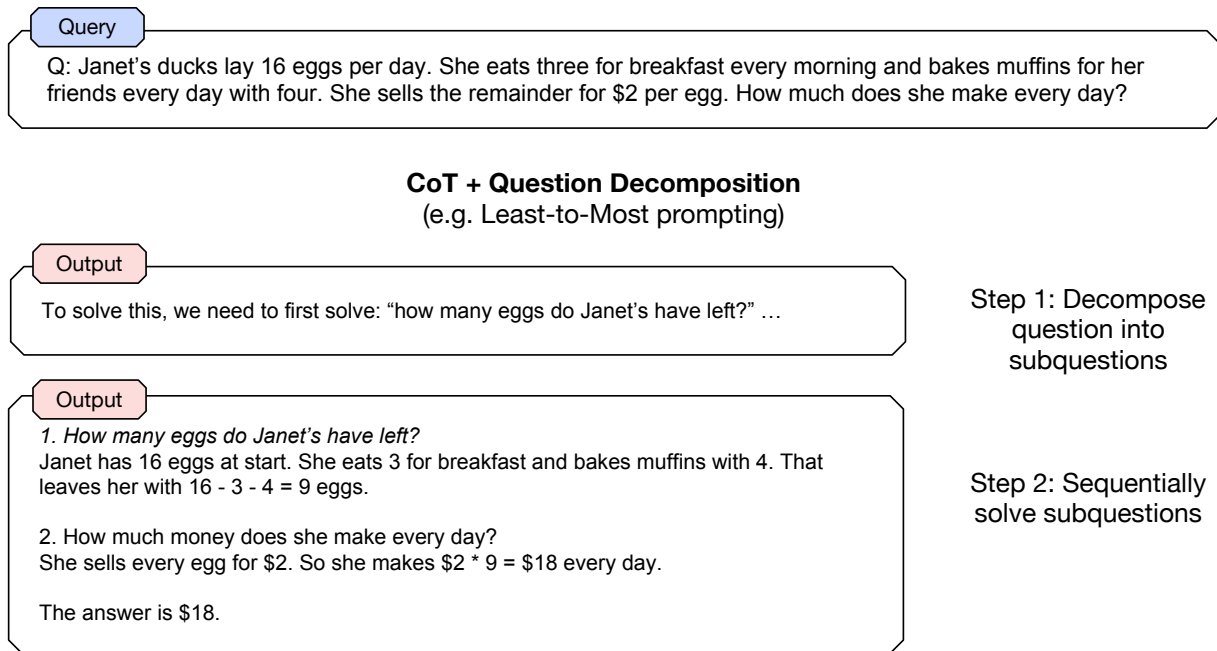


Figure 2.3: An example output from CoT prompting with question decomposition (e.g. Zhou et al., 2022).

answers, as in Self-Consistency CoT (Wang et al., 2022b) (Figure 2.4), or voting on the level of steps, as in DIVERSE (Li et al., 2022). These vote-and-rank techniques can in theory be applied to any previously mentioned methods, and have been shown to consistently improve their performance on multiple end tasks.

Despite significant advancements, we contend that the current methodology of generating free-text explanations using LLMs raises critical concerns regarding both **faithfulness** and **utility**.

First, there is no theoretical assurance that LLM-generated explanations are faithful, i.e., they accurately describe the reasoning mechanism behind the model's prediction. This is because LLMs generate texts by sampling tokens from the vocabulary according to a probability distribution. Thus, there is no intrinsic constraint to ensure that the reasoning process described in the explanation is the same as the one used to produce the prediction. Empirical research has demonstrated that LLMs can produce explanations that are *systematically unfaithful* (Turpin et al., 2023). When LLMs rely on implicit biases to make predictions, they still generate explanations that look plausible to humans, hiding these biases. Furthermore, larger models are particularly prone to generating

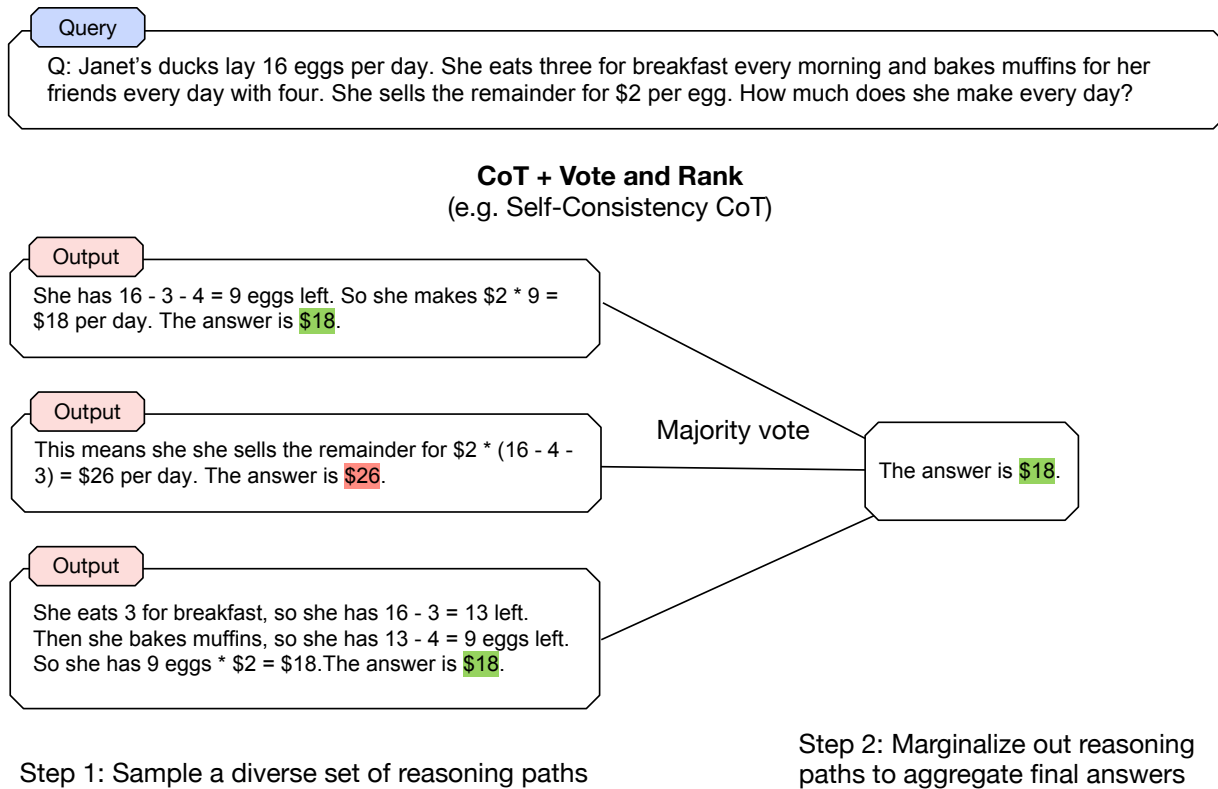


Figure 2.4: An example output from CoT prompting with vote-and-rank (e.g. Wang et al., 2022b).

unfaithful explanations (Lanham et al., 2023). This lack of faithfulness can lead to a misleading perception of “self-interpretability” and result in increased over-reliance on the model, especially when the explanations appear convincing (Slack et al., 2020).

Second, studies have indicated that LLM-generated explanations often lack practical utility. For example, in the context of collaborative decision-making, these explanations can be *adversarially helpful* (Ajwani et al., 2024). This appears when explanations make incorrect answers look plausible, potentially leading users to trust them. Additionally, in the scenario of knowledge discovery, only a small proportion (approximately 20%) of LLM-generated explanations prove to be useful in aiding lay people to answer previously unseen questions (Joshi et al., 2023). This uncertainty in usefulness undermines the fundamental purpose of explanations, which is to help the target audience understand and effectively apply the model’s outputs in real-world situations.

2.3.2. Structured Explanations

In contrast to free-text explanations, structured explanations involve a non-linear (e.g., tree-like or graph-like) and less open-ended mode of reasoning. This is because many reasoning problems require exploration and backtracking (e.g., planning, multi-hop QA), and/or reasoning under certain constraints (e.g., logical deduction). These issues can be addressed by integrating explanations with structures, such as logical constraints, symbolic programs, and non-linear exploration strategies.

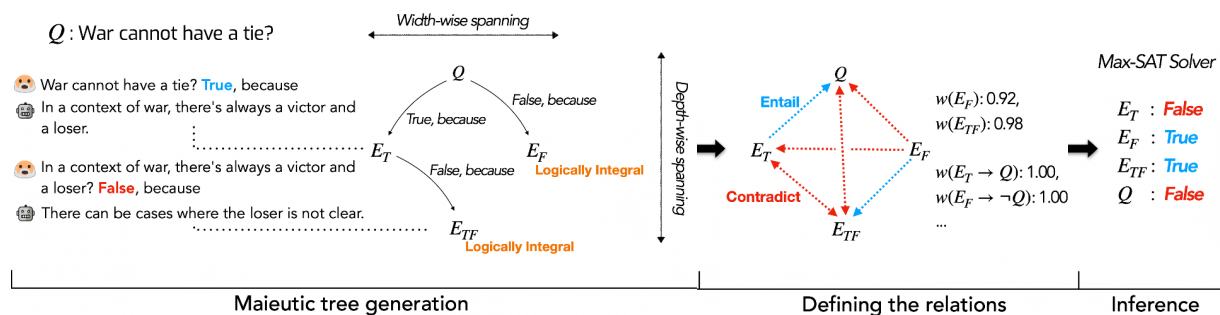


Figure 2.5: An example output from one type of structured explanations with logical constraints — Maieutic prompting (figure from Jung et al. (2022a)).

Logical constraints provide a framework for reasoning that adheres to formal rules of inference. In the context of structured explanations, these constraints ensure that the explanation follows a logically sound path, maintaining consistency and validity throughout the reasoning process. For example, Maieutic Prompting in Figure 2.5 (Jung et al., 2022b) prompts the LLM to generate a *maieutic tree*, which consists of abductive and recursive explanations as well as the logical relation among them. Each node represents a proposition, and each edge stands for a weighted logical relation (entailment, contradiction, neutral) between two propositions. Then, each proposition is assigned a truth value, such that the sum of the weights of satisfied relation constraints is maximized. Similarly, SatLM (Ye et al., 2024) prompts the LLM to generate a declarative task specification as a set of logical constraints. Then, it calls an off-the-shelf automated theorem prover to derive the final answer. By incorporating these constraints, explanations become more rigorous and verifiable, which are particularly effective on multi-hop QA, arithmetic reasoning, and logical reasoning tasks.

Symbolic programs offer a powerful means of representing and manipulating abstract entities

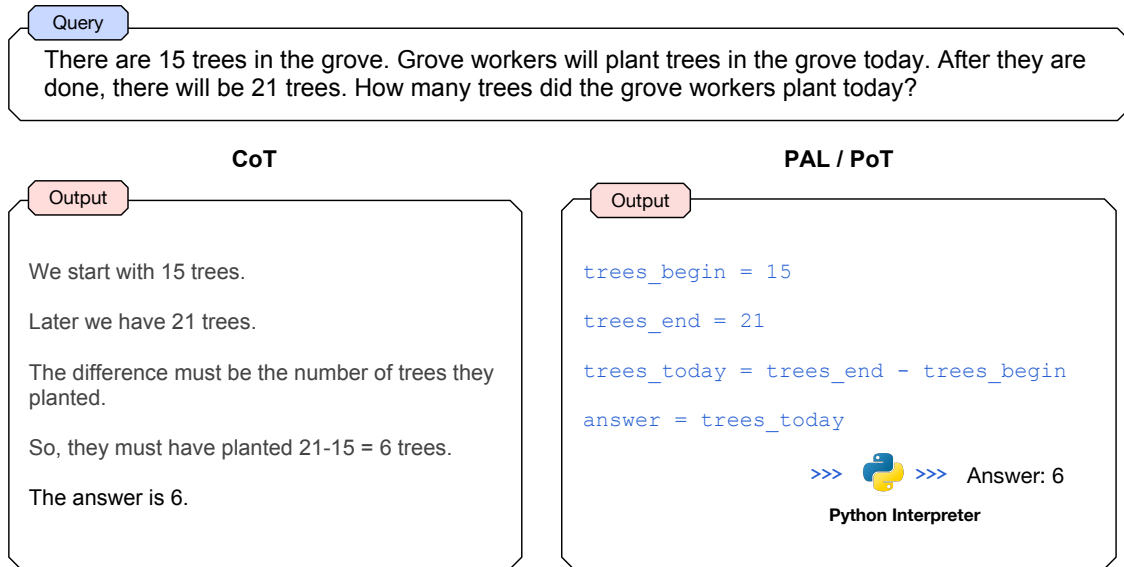


Figure 2.6: An example output from two types of explanations with symbolic programs — Program of Thought (PoT) (Chen et al., 2022b) and Program-Aided LMs (PAL) Gao et al. (2022).

and operations deterministically. Two concurrent approaches illustrated in Figure 2.6, Program-of-Thought (PoT) (Chen et al., 2022b) and Program-Aided Language Models (PAL) (Gao et al., 2022), both prompt the LLM to generate reasoning chains in the format of a Python program and obtain the final answer by executing the program with a deterministic Python interpreter. This process essentially offload the answer derivation process to the dedicated program executor, allowing the LLM to solely focus on generating the symbolic representation of the problem. Compared with free-text explanations, this approach leads to better performance on multiple arithmetic and symbolic reasoning tasks.

Non-linear exploration strategies is designed to address the limitation that chain-like structures can limit the scope of exploration, thus making it difficult to recover from errors in the reasoning path. Instead of following a single, straightforward trajectory, non-linear strategies allow for backtracking, branching, and revisiting previous steps as necessary. As shown in Figure 2.7, Tree of Thoughts (ToT) (Yao et al., 2024) extends standard CoT by allowing the model to consider multiple branches of reasoning paths. It determines which ones to pursue by self-evaluating each branch, as well as looking ahead or backtracking when necessary. On top of that, Graph of Thoughts (Besta et al., 2024) further

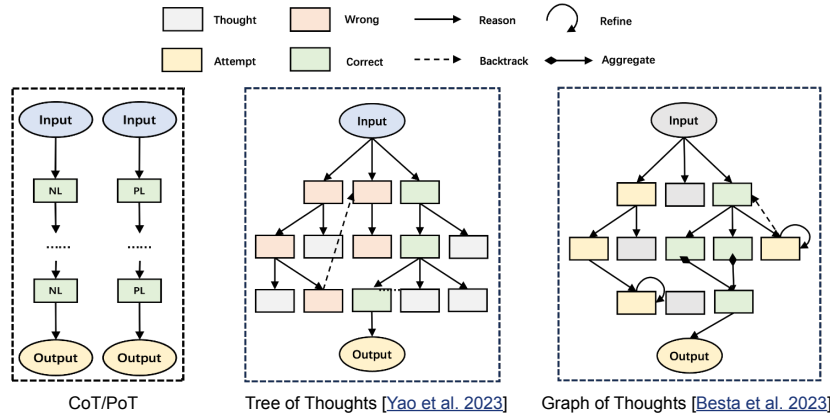


Figure 2.7: An example output from two types of explanations with non-linear exploration strategies — Tree of Thoughts (Yao et al., 2024) and Graph of Thoughts (Besta et al., 2024). Figure from Chu et al. (2023).

introduces loops and N-to-1 connections between steps, which improves subproblem aggregation and self-verification. These methods are particularly relevant in problem-solving scenarios where the optimal solution path is not immediately apparent, such as in planning and strategic games (e.g., Crosswords and Game of 24).

Work in this thesis, in particular Chapter 4 on Faithful CoT, is one instance of integrating symbolic programs with explanations. We highlight a few key differences between our work and concurrent work on the same topic (Gao et al., 2022; Chen et al., 2022b): (a) In our approach, we first decompose the original problem into a dependency graph of subproblems and then generate symbolic programs to solve each subproblem, effectively reducing the task difficulty for the LLM at each stage; in contrast, concurrent studies directly generate symbolic program to solve the original problem. (b) We demonstrate the generalizability of our approach to multiple symbolic languages beyond Python and multiple domains beyond arithmetic reasoning and simple symbolic reasoning. In particular, we innovatively recast a diverse set of realistic tasks (Planning, Multi-hop QA, and Relational Inference) into a symbolic representation, which allows us to tackle them with a single framework. (c) Empirically, our approach performs better than concurrent approaches on knowledge-intensive tasks (Appendix A.1).

2.4. Summary

In this chapter, we have discussed the foundational concepts pertinent to the theme of this thesis, namely LLMs, explainability, and LLM-generated explanations. We began by discussing the evolution and significance of LLMs, highlighting their role in various NLP tasks and the transition through different paradigms of model development.

We then delved into the theoretical concept of explainability, emphasizing its importance in making AI systems more reliable and trustworthy. We discussed various properties and principles that constitute effective explanations, laying the groundwork for assessing the faithfulness and utility of explanations produced by LLMs.

Finally, we reviewed recent advances in LLM-generated explanations, identifying critical gaps in faithfulness and utility. This review highlighted the challenges and opportunities in leveraging LLMs to generate explanations that are both reliable and beneficial. By identifying these gaps, we set the stage for the subsequent chapters, where we propose and evaluate novel methods to enhance the faithfulness and utility of LLM-generated explanations.

This chapter’s discussion provides the necessary background to understand the complexities involved in generating faithful and useful explanations by LLMs. It also establishes the context for the taxonomy and methodological contributions that follow. In the next chapter, we will present a detailed taxonomy of model explanation methods in NLP, categorizing existing approaches and situating LLM-generated explanations within this broader landscape.

CHAPTER 3

A Taxonomy of Model Explanation Methods in NLP

Building upon the foundational concepts discussed in the Chapter 2, we now turn our focus to categorizing and examining various model explanation methods in NLP.

In this chapter, we provide a taxonomy to categorize existing model explanation methods in NLP into five families: similarity-based methods, analysis of model-internal structures, backpropagation-based methods, counterfactual intervention, and self-explanatory models. A considerable amount of this chapter is based on our survey article titled “Towards Faithful Model Explanation in NLP: A Survey” (Lyu et al., 2022a). This article also includes a deep dive into the concept of faithfulness, a critical review of faithfulness evaluation methods, and a discussion on challenges and opportunities in explainable NLP from the perspective of faithfulness, omitted for brevity.

Method	Time	Model accessibility	Scope	Unit of explanation	Form of explanation
Similarity-based methods	post-hoc	white-box	local	examples, concepts	importance scores
Analysis of model-internal structures	post-hoc	white-box	local, global	features, interactions	visualization, importance scores
Backpropagation-based methods	post-hoc	white-box	local	features, interactions	visualization, importance scores
Counterfactual intervention	post-hoc	black-box, white-box	local, global	features, examples, concepts	importance scores
Self-explanatory models	built-in	white-box	local, global	features, examples, concepts	importance scores, natural language, causal graphs

Table 3.1: Comparison of different model explanation methods in terms of their properties. Different colors denote different values of a property. See Section 2.2.2 for details.

Our taxonomy is driven by the properties of explanations discussed in Section 2.2.2. As a preview, Table 3.1 compares these methods to be discussed in terms of the properties.

Note that many existing taxonomies do not explicitly state which properties are taken into account, thus often producing confusing terms. For instance, the term *saliency methods* has been widely used to refer to “backpropagation-based methods” in our taxonomy. However, *saliency* only describes the form and the unit of explanation – importance scores of input features. Then, technically speaking, all methods in Table 3.1 have instances that can be called a *saliency method*. In our taxonomy, we characterize each method using all these properties, so as to provide a clearer, mutually exclusive distinction.

Next, we will introduce the five method families in detail. We will illustrate each of them with representative studies, and discuss their key strengths and weaknesses.⁸ At the end of this chapter, we will situate LLM-generated explanations within this taxonomy, laying the groundwork for the novel frameworks and techniques proposed in the following chapters of this thesis.

3.1. Similarity-based methods

Similarity-based methods provide explanations in terms of training examples. Specifically, in order to explain the model prediction on a test example, they find its most similar⁹ training examples in the learned representation space, as support for the current prediction. This is akin to how humans explain their actions by analogy, e.g., doctors make diagnoses based on past cases.

Caruana et al. (1999) theoretically formalize the earliest approach of this kind, named “case-based explanation”. Based on the learned hidden activations of the trained model, it finds the test example’s k-Nearest Neighbors (kNN) in the training set as an explanation. Note that the similarity is defined in terms of the *model’s learned representation space* but not the *input feature space*, since otherwise the explanation would be model-independent.

Wallace et al. (2018) also use the kNN search algorithm; but instead of deriving a post-hoc explanation, they replace the model’s final softmax classifier with a kNN classifier at test time. Concretely, during training, the model architecture is unmodified. Then, with the trained parameters fixed,

⁸Due to space considerations, the content has been significantly abridged; please see our survey (Lyu et al., 2022a) for a more comprehensive version.

⁹In practice, commonly used similarity metrics include cosine, Euclidean, etc.

each training example is passed through the model again, and their representations are saved. The inference is done with a modified architecture: a test example is classified based on the labels of its kNNs from the training examples in the learned representation space. However, the resulting explanations are only evaluated based on whether the explanations align with human perception of feature importance on qualitative examples, which is irrelevant to faithfulness.

Rajagopal et al. (2021) introduce a self-explanatory classification model, where one component, the “global interpretable layer”, also uses the idea of similarity-based explanation. This layer essentially identifies the most similar *concepts* (phrases in this case) in the training data for a given test example. Their approach is mainly evaluated in terms of plausibility, i.e., how adequate/understandable/trustworthy an explanation is based on human judgment. One metric touches on faithfulness — human simulatability — however, the authors only report the relative difference with and without the explanation instead of the absolute scores, which makes it hard to determine how faithful the approach is.

3.1.1. Strengths and Weaknesses

Similarity-based methods exhibit several strengths. First, they are **intuitive** to humans since the justification by analogy paradigm has long been established. Second, they also **easy to implement**, as no re-training or data manipulation is needed. The similarity scores are available by simply passing examples through the trained model to obtain the model’s representation of them. Third, they are **highly model-agnostic**, since all kinds of Neural Networks have a representation space, and any similarity metric (cosine, Euclidean, etc.) can be easily applied. Finally, certain similarity-based explanations are rated by human subjects as more **understandable and trustworthy** (Rajagopal et al., 2021) compared to several other baselines in the families of backpropagation-based methods and counterfactual intervention.

Nevertheless, there are also several weaknesses: First, most similarity-based methods only provide the user with the **outcome** of the model’s reasoning process (i.e., which examples are similar in the learned space), but do not shed light on *how* the model reasons (i.e., how the space is learned) (Caruana et al., 1999). Second, existing work mostly evaluates the resulting explanations with

plausibility-related metrics, including adequacy, relevance, understandability, etc., with human judgments. However, plausibility does not entail **faithfulness**, as models do not necessarily use a similar reasoning mechanism as humans do. Thus, it is questionable whether similarity-based methods can truly establish causality between model predictions and the explanation. Additionally, the space of exploration is **confined to the training set**. This inherently limits the diversity and scope of the explanation, potentially leaving certain edge cases unexplained. It also implies that the behavior of similarity-based methods depends on the distribution of the training data. In other words, the explanation outcomes may vary considerably if the training set does not well represent the broader data distribution or is biased in some manner. Finally, similarity-based methods inherently offer **instance-level explanations** and do not provide insights into the feature-level contribution to the prediction. This lack of granularity can limit the potential for actionable insights.

3.2. Analysis of model-internal structures

The analysis of model-internal structures, such as individual neurons and mechanisms like convolution or attention, aims to elucidate the inner workings of NLP models. Common techniques include visualization (e.g., activation heatmaps), clustering (e.g., neurons with similar functions), and correlation analysis (e.g., between neuron activations and linguistic properties).

3.2.1. Analysis on Neurons

Early interest in neural models in NLP focused on finding interpretable functions of individual neurons. Karpathy et al. (2015) examined neuron activation patterns in a character-level LSTM language model, identifying neurons that activate within quotes or specific syntactic structures. Subsequent studies by Li et al. (2016a) and Strobelt et al. (2018) found neurons in LSTMs that respond to language compositions such as negation and adjective-noun pairs. Conversely, Poerner et al. (2018) and Hiebert et al. (2018) explored inputs that produce similar neuron activations, showing that models like GRUs and LSTMs capture orthographic and grammatical patterns.

3.2.2. Analysis on Attention Mechanism

The Transformer architecture (Vaswani et al., 2017), particularly its self-attention mechanism, has become foundational in modern NLP. Self-attention assigns weights to input vectors, which are

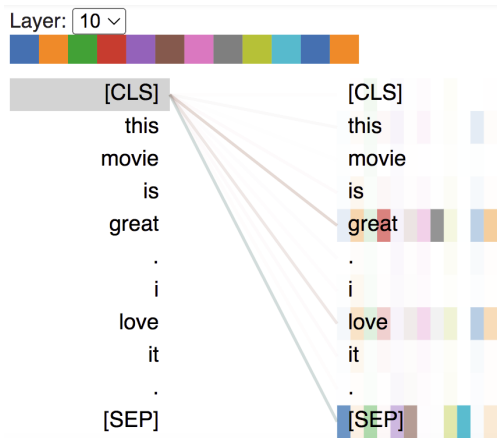


Figure 3.1: Attention weights from [CLS] to all tokens in a BERT sentiment classification model, created using BertViz Vig (2019). Each color represents an attention head. Lines represent averaged attention weights over all heads. Darker shades indicate higher weights.

then used to compute weighted averages, producing composite embeddings. This mechanism is often interpreted as highlighting the importance of input tokens to the output. For example, in a BERT-based classifier shown in Figure 3.1, the final “[CLS]” token might show high attention weights for tokens like “great” and “love” in a movie review, which can be understood as the most important features supporting the positive sentiment classification.

However, there has been a long debate on whether such explanations are faithful. Jain and Wallace (2019) demonstrated that adversarial attention distributions could significantly alter attention weights without changing model predictions, questioning the causal link between attention and output. Wiegrefe and Pinter (2019) countered that adversarial distributions are not naturally learned by models, suggesting that models typically do not focus on uninformative words. Pruthi et al. (2020) further refuted this by showing that models could be trained to produce misleading attention weights intentionally.

To improve the faithfulness of attention, researchers have proposed various methods. Tutek and Snajder (2020) introduced regularization techniques to make hidden states more representative of input features. Abnar and Zuidema (2020) proposed Attention Rollout and Attention Flow to capture global information flow, showing increased plausibility and faithfulness. Additionally, tying attention to other explanation methods, especially backpropagation-based ones, has been suggested to enhance

causality (Hao et al., 2021; Lu et al., 2021).

Despite the controversy on faithfulness, recent cognitive science studies show that attention mechanisms in models share similarities with human brain activation patterns during tasks like masked word prediction and eye fixation during reading, indicating a convergence between how models and humans process language.

3.2.3. Strengths and Weaknesses

To summarize, the analysis of model-internal structures, as a family of explanation methods, has several strengths. First, the visualization of model-internal structures is **intuitive and readable** to humans, especially end-users. Second, the attention mechanism can capture the **interaction** between features, whereas many other methods can only capture the influence of individual features themselves. Third, model weights are **easily accessible and computationally efficient**, compared to other methods.

However, these methods also suffer from several key weaknesses: First, it is questionable to what extent raw attention weights represent **causal contribution**, as mentioned in the debate on the faithfulness of attention as explanation. Second, the lack of faithfulness may be due to our interpretation of attention weights on intermediate **hidden states** as the importance of input features; however, hidden states have already mixed in contextual information through previous self-attention layers, and therefore may not be representative of input features. Finally, existing methods often focus on attention weights in a single layer and/or from a single token position. This may reflect how much the model attends to each input position **locally**, but without taking the whole computation path into account. Methods that characterize the global information flow may be a better alternative (Abnar and Zuidema, 2020, i.a.).

3.3. Backpropagation-based methods

Backpropagation-based methods aim to identify the contribution of input features via a *backward pass* through the model, propagating the *importance* (or *relevance*, used interchangeably in the literature) attribution scores from the output layer to the input features. They can be further distinguished

into two categories, **gradient methods** and **propagation methods**. Gradient methods follow standard backpropagation rules. In other words, they directly compute the *gradient* (or some variant of it) of the output w.r.t the input features via the chain rule, assuming features with larger gradient values are more influential to the model prediction. By contrast, propagation methods define custom backpropagation rules for different layer types and compute the relevance scores layer by layer until reaching the input. This is believed to better capture the redistribution of relevance through special layers, such as ReLU.

Most ideas in this family have been first proposed in Computer Vision (CV). In the following subsections, we will explain their origin in vision and adaptation to language.

3.3.1. Gradient methods

Gradient methods compute the gradient of the model output with respect to each input feature, assuming features with larger absolute gradient values are more influential. The simplest approach is to use the gradient itself (referred to as **Simple Gradients** or **Vanilla Gradients**) (Baehrens et al., 2010; Simonyan et al., 2014). The gradient’s sign indicates whether a feature contributes positively or negatively, and its magnitude indicates the extent of influence. However, Vanilla Gradients suffer from issues like saturation. For common neuron activation functions like sigmoid ($y = \frac{1}{1+e^{-x}}$) and tanh ($y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$), when the absolute value of an input feature is large enough, it results in a very small gradient locally, although the feature may have a large contribution to the output y globally.

Gradient×Input (Denil et al., 2015) was proposed to address this issue by computing the relevance score as the product of the input feature and its gradient, analogous to linear models. **Integrated Gradients** (Sundararajan et al., 2017) takes a different approach to estimate the global relevance of a feature by comparing the input with a baseline input (e.g., an all-black image for vision and a sentence with all-zero token embeddings for language). However, this method can still result in visually noisy relevance maps in CV. **SmoothGrad** (Smilkov et al., 2017) aims to reduce this noise by averaging the relevance maps of noisy copies of the input, improving visual clarity empirically. Nevertheless, this only pertains to plausibility, but not faithfulness.

In NLP, both Simple Gradients and Integrated Gradients have been used, primarily for sequence classification tasks. For instance, Li et al. (2016a) used Simple Gradients to explain token importance in RNN models, while Hao et al. (2021) and Janizek et al. (2021) adapted Integrated Gradients for Transformer models on tasks like paraphrase detection and sentiment classification.

3.3.2. Propagation methods

Propagation methods use custom backward passes with specific rules for different layers. For example, **DeconvNet** (Zeiler and Fergus, 2014) and **Guided BackPropagation** (GBP) (Springenberg et al., 2015) define custom rules for ReLU units, zeroing out relevance only under certain conditions. These methods produce clearer visualizations than Vanilla Gradients. However, it is shown that both methods are essentially doing (partial) input recovery, which is unrelated to the network’s decision (Nie et al., 2018).

Layerwise Relevance Propagation (LRP) (Bach et al., 2015) generalizes this approach with a high-level Relevance Conservation constraint, ensuring total incoming relevance equals total outgoing relevance for each neuron. LRP has shown improved faithfulness in perturbation-based evaluations, such as token deletion tests in NLP tasks. However, it still suffers from the same saturation issue mentioned in Section 3.3.1.

To address LRP’s failure with saturation, two *reference-based* methods, **DeepLift** (Shrikumar et al., 2017) and **Deep-Taylor Decomposition (DTD)** (Montavon et al., 2017), have been introduced. Analogous to Integrated Gradients, they aim to measure the *global* contribution of input features by finding a reference point, or baseline, to compare with the input. The two methods differ in how the baseline input is chosen. DeepLift empirically chooses a baseline input which results in a neutral output, but DTD additionally requires the baseline to lie in the vicinity of the original input. Both methods avoid saturation issues. In NLP, Chefer et al. (2021) extend DTD to explain the decision of Transformer models on sentiment classification. However, the explanations are evaluated against human-annotated token relevance, therefore also unrelated to faithfulness.

3.3.3. Strengths and Weaknesses

Summarizing the discussion above, backpropagation-based methods have several key strengths. First, they generate a spectrum of feature relevance scores, which is easily **understandable** for all kinds of target users. Second, the **computational cost** of these methods can vary significantly, but in general, they are relatively efficient to compute. Gradient-based techniques only require a handful of calls to the model’s backward function. On the other hand, propagation methods involve a customized implementation of the backward pass, allowing for precise control over the relevance redistribution process while necessitating more complex computation.¹⁰ Third, in terms of **faithfulness**, gradients (and variants) are intrinsically tied to the influence of input features on the prediction. Empirically, certain recently proposed methods (e.g., Layerwise Relevance Propagation, DeepLift, Deep-Taylor Decomposition) are shown to be more faithful than previous baselines via perturbation-based evaluation, as mentioned before. Finally, unlike most methods for the analysis of model-internal structures (e.g., raw attention weights), backpropagation-based methods take the **entire computation path** into account.

At the same time, these methods are far from perfect due to a number of weaknesses. First, most existing backpropagation-based methods target **low-level features** only, e.g., pixels in vision and input tokens in language. It is unclear how to compute any sort of gradient w.r.t. higher-level features like case, gender, part-of-speech, semantic role, syntactic dependency, coreference, discourse relations, and so on. Second, it is questionable how to apply such methods to **non-classification tasks**, especially when there is no single output of the model, e.g., text generation or structured prediction. Additionally, as detailed before, certain methods **violate axiomatic principles** of faithfulness, e.g., Input Sensitivity and Model Sensitivity (Sundararajan et al., 2017). Lastly, the explanation can be **unstable**, i.e., minimally different inputs can lead to drastically different relevance maps (Ghorbani et al., 2019; Feng et al., 2018). Most methods are **not empirically evaluated on faithfulness** when they are first proposed, with only a few exceptions mentioned above. Moreover,

¹⁰In general, their computational cost is lower than counterfactual intervention, which typically requires multiple forward passes in addition, but higher than internal-structure analysis, which typically requires no additional model calls. It’s worth noting that these computational costs exist along a continuum rather than being binary categories.

subsequent researchers find many systematic deficiencies of them in ad-hoc evaluations. As mentioned before, Guided BackPropagation and DeconvNet are shown to be only doing partial input recovery, regardless of the model’s behavior (Nie et al., 2018). In addition, certain explanations (including Simple Gradients, Integrated Gradients, and SmoothGrad) can be adversarially manipulated, i.e., one can construct entirely different gradient distributions with little influence on the prediction (Wang et al., 2020).

3.4. Counterfactual intervention

The notion of counterfactual reasoning stems from the causality literature in social science: “Given two co-occurring events A and B , A is said to cause B if, under some hypothetical counterfactual case that A did not occur, B would not have occurred” (Roese and Olson, 1995; Winship and Morgan, 1999; Lipton, 2016). In the context of machine learning, counterfactual intervention methods explain the causal effect between a feature/concept/example and the prediction by *erasing* or *perturbing* it and observing the change in the prediction. A larger change indicates stronger importance.

One axis along which we can categorize existing studies is *where the intervention happens*, in **inputs** or in **model representations**. The former manipulates the input and passes it through the original model; by contrast, the latter directly intervenes in the model-internal representations, e.g., neurons or layers. The rest of this section will elaborate on the two categories accordingly.

3.4.1. Input Intervention

Input intervention methods can be further categorized along two dimensions: the *target* and the *operation*. The target refers to “what is affected by the intervention”, normally input **features** (e.g., tokens) or **examples** (the entire input instance). The operation stands for the specific intervention method, which can be **erasure** (masking out or deleting the target) or **perturbation** (changing the value of the target). We will first classify existing work based on the target, and then on the operation.

Feature-targeted Intervention

Early feature-targeted methods mostly relied on *erasure*. One common approach is **leave-one-out** (Li et al., 2016b), which erases single features (e.g., tokens) and assesses the resulting change in prediction. However, this method captures only the linear contribution of single features and cannot handle higher-order interactions.

To address this, some methods erase subsets of features. For example, Li et al. (2016b) aimed to find the minimum subset of input tokens to erase to flip the model’s decision. Alternatively, Ribeiro et al. (2018) sought to find the minimum subset of tokens to keep the decision unchanged, known as "Anchors". Both methods rely on approximate searches due to the intractability of finding exact subsets. Surrogate models offer another approach. **LIME** (Local Interpretable Model-agnostic Explanations) (Ribeiro et al., 2016) approximates a black-box model locally with an interpretable model. **SHAP** (SHapley Additive exPlanations) (Lundberg and Lee, 2017) uses additive surrogate models based on game theory to explain feature contributions. LIME is evaluated for faithfulness through white-box tests, and SHAP is theoretically locally faithful.

A common problem with all the above feature erasure methods is that they can produce out-of-distribution (OOD) inputs, for example, ungrammatical or nonsensical sentences after tokens are masked out. Exploiting this weakness, Slack et al. (2020) show that it is possible to *fool* popular erasure-based explanation methods with an adversarial model, which relies on biased features on in-distribution inputs and unbiased ones on OOD inputs.

This motivates the other operation – **perturbation** – as another type of feature-targeted intervention. Compared to simple erasure, perturbing the value of the feature is less likely to result in OOD inputs. The outcome of such perturbations is called *counterfactual examples*, which should be meaningfully different from the original while being semantically coherent.

Generating high-quality counterfactual examples is non-trivial, as they need to simultaneously accord with the counterfactual target label, be semantically coherent, and only differ from the original example in the intended feature. The most reliable (yet expensive) approach to collect

counterfactual examples is still manual creation, as in Kaushik et al. (2020) and Abraham et al. (2022). However, recent studies propose promising ways to automate their generation (Calderon et al., 2022; Zmigrod et al., 2019; Amini et al., 2022; Wu et al., 2021).

Example-targeted Intervention

In addition to feature-targeted intervention, counterfactual input intervention can also directly happen on the level of examples. A representative method is called **influence functions** (Koh and Liang, 2017), which is designed to explain which training examples most influence the prediction of a test example. This may remind us of similarity-based methods in Section 3.1. Though sharing the same unit of explanation (training instances), similarity-based methods rely on similarity search, while influence functions rely on counterfactual reasoning by *approximating the change in prediction* if a training example were absent or slightly changed. This method has been adapted to NLP (Han et al., 2020) after its invention in CV, but the approximation’s accuracy can vary, raising concerns about reliability (Basu et al., 2021).

3.4.2. Model Representation Intervention

Similar to input intervention methods, we also categorize model representation intervention methods according to the *target* and the *operation*. Here, the target can typically be individual **neurons** or high-level **feature representations**. The operation still involves **erasure** and **perturbation**. We will still introduce existing work along the line of target first and operation next.

Neuron-targeted Intervention

By intervening in individual neurons in an NN, one can explain the importance of each neuron to the prediction. The intervention can still be either **erasure** or **perturbation**.

The simplest form of erasure is still **leave-one-out**. Using the same strategy as with input features, Li et al. (2016b) study the effect of zeroing out a single dimension in hidden units on the prediction. Bau et al. (2019) improve the efficiency of this method, by searching for important neurons in a guided fashion instead of brute-force enumeration.

Apart from erasure, perturbation is another form of neuron-targeted intervention. One representative

example is **causal mediation analysis** (Vig et al., 2020), which measures the effect of a control variable (input) on a response variable (output) mediated by an intermediate variable (neuron). This framework has been used to analyze biases and linguistic phenomena in models (Vig et al., 2020).

Feature-representation-targeted Intervention

Beyond intervening in neurons, directly targeting feature representations in the model allows us to answer more insightful questions like “Is some high-level feature, e.g., syntax tree, used in prediction?”.

Similar to neuron-targeted intervention, the most intuitive way to perform an intervention on feature representation is **erasure**. Two pieces of concurrent work, **Amnesic Probing** (Elazar et al., 2021) and **CausalLM** (Feder et al., 2021), are representative examples, erase target features from model representations to measure their impact on predictions. They exploit different algorithms to erase the target feature from the model representation, via either Iterative Null-space Projection (INLP) (Ravfogel et al., 2020) or adversarial training. However, both face challenges in that they cannot guarantee to remove the target feature entirely; even worse, they may end up destroying other task-relevant features.

Taking a step back, even with perfect erasure techniques, a higher-level problem with the feature representation erasure methodology lies in unrealistic representations, similar to OOD inputs in the case of input erasure.

To address this issue, **perturbation**-based methods targeting feature representations are proposed (Ravfogel et al., 2021; Tucker et al., 2021). Instead of removing the feature altogether, they manipulate the feature value, such that the same feature (e.g. syntax tree) is represented counterfactually (e.g. with different structures) in the model. For example, a syntactically ambiguous sentence such as *I saw the boy and the girl [MASK] tall.* can be interpreted as either *[I saw the boy] and [the girl [MASK] tall].*, or *I saw [the boy and the girl [MASK] tall].* Therefore, *[MASK]* should be either singular or plural, if the model relies on syntax tree in word prediction.

3.4.3. Strengths and Weaknesses

Overall, there are several advantages unique to counterfactual intervention methods. First, having its root in the causality literature, counterfactual intervention is therefore designed to **capture causal instead of mere correlational effects** between inputs and outputs. Second, compared to other methods, counterfactual intervention methods are more often explicitly evaluated in terms of **faithfulness** (e.g., Ribeiro et al., 2018; De Cao et al., 2020; Ribeiro et al., 2016; Lundberg and Lee, 2017; Tsang et al., 2020; Feder et al., 2021), and often outperform existing baselines.

However, they also share a number of disadvantages. First, compared to other methods, counterfactual intervention is relatively more expensive in **computational cost**, normally requiring multiple forward passes and modifications to the model representation. Second, as explained before, erasure-based intervention can result in **nonsensical inputs or representations**, which sometimes allow adversaries to manipulate the explanation (Slack et al., 2020). Third, intervening in a single feature relies on the assumption that features are **independent**, failing to capture feature interactions. Additionally, interventions are often overly **specific to the particular example** (Wallace et al., 2020), lacking broader insights about the model. Finally, counterfactual intervention may suffer from **hindsight bias** (De Cao et al., 2020), which questions the foundation of counterfactual reasoning. Specifically, the fact that a feature can be dropped without influencing the prediction *might not mean* that the model “knows” that it can be dropped and has not used it in the original prediction.¹¹

3.5. Self-explanatory Models

In contrast with all the above post-hoc methods, self-explanatory models provide built-in explanations. Typically, explanations can be in the form of feature importance scores, natural language, causal graphs, or the network architecture itself.

Prior work on self-explanatory models can be broadly categorized into two lines based on *how the*

¹¹De Cao et al. (2020) illustrates this point with an intuitive example of the Reading Comprehension task, where a model is given a paragraph and a question, and should identify an answer span in the paragraph. Now, using counterfactual intervention, if we mask out everything except the answer in the paragraph, the model will for sure predict the gold span. Nonetheless, this does not imply that everything else is unimportant for the model’s original prediction. The issue again calls for a rethinking of the fundamental assumptions of counterfactual reasoning.

explanation is formed: **explainable architecture** or **generating explanations**. The former relies on a transparent model architecture, such that no extra explanation is necessary. The latter, though, may still involve opaque architectures, but generates explicit explanations as a byproduct of the inference process.

3.5.1. Explainable Architecture

Explainable architectures are designed to be inherently interpretable, drawing inspiration from classic machine learning models like decision trees and linear regression.

Neural Module Networks (NMNs) are one representative example, specifically in the context of Question Answering (QA) tasks. They decompose a complex question into a sequence of steps or modules. For example, NMNs parse the input question into a program of learnable modules, which are then executed to derive the answer (Andreas et al., 2016; Hu et al., 2017). This approach has been effective in various visual QA tasks and has been extended to language-only domains, although challenges remain in ensuring the modules’ behavior aligns with their intended functions (Gupta et al., 2020). Despite their transparency, NMNs face faithfulness issues. The modules may not perform their intended functions, as their parameters are often learned end-to-end, which may not reflect the actual reasoning process (Subramanian et al., 2020). Additionally, symbolic modules may not capture the flexible semantics of natural language, limiting their applicability to complex tasks.

Models with constraints are another example of explainable architectures. The idea is to incorporate constraints into neural networks from classic interpretable models, like generalized linear regression (Alvarez-Melis and Jaakkola, 2018) and finite-state automata (Schwartz et al., 2018; Deutsch et al., 2019; Jiang et al., 2020). Still, a major challenge lies in the trade-off between interpretability and performance.

3.5.2. Generating Explanations

Besides using architecture as an implicit explanation, another type of self-explanatory model learns to generate an explicit explanation as an additional task, aside from making the prediction. This

line of work is the most relevant to the research presented in my thesis.

To supervise self-explanatory models, human-written explanations are often used as an additional training signal, along with the end-task label. According to the dependency relationship between the *predictor* and the *explainer*, we can classify existing work into three categories: **predict-then-explain**, **explain-then-predict**, and **jointly-predict-and-explain**.

Predict-then-explain

In this framework, a standard predictor first makes a prediction, which is then justified by an explainer. This is analogous to previous post-hoc explanation methods. While widely used, it suffers from the same faithfulness challenge as all other post-hoc methods: since the predictor does not depend on the explainer, there is no guarantee that the explanation accurately reflects the reasoning process behind the prediction. Moreover, as the supervision comes from human-provided explanations, the explainer is only explicitly optimized in terms of plausibility, but not faithfulness.

Explain-then-predict

Here, the explainer generates an explanation first, which then serves as the **only input** for the predictor. This approach aims to make the prediction faithful by construction since the prediction relies *solely* on the explanation. One common method is extractive rationales, where a part of the input sufficient for prediction is extracted and used (Lei et al., 2016; Bastings et al., 2019). Although rationale-based models seem to be “faithful by construction”, they are not necessarily so. The predictor might only be exploiting superficial patterns in the rationale (e.g., the number of tokens that are kept (Jacovi and Goldberg, 2021)).

Alternative to extracted rationales, a more flexible form of explanation is natural language (also called *free-text* explanation). Similarly, the explanation might contain cues to the label. For example, in an NLI task, patterns like *X implies Y* / *X is a type of Y* in the explanation oftentimes indicate **Entailment**, while *X is not the same as Y* is a strong signal of **Contradiction**. To overcome this issue, Kumar and Talukdar (2020) propose the Natural language Inference over Label-specific Explanations (NILE) model, where every class label has a corresponding explainer. All three

explanations are fed to the predictor, which makes a decision after comparing them.

Jointly-predict-and-explain

This approach can have two structures: (i) there are still an explainer and a predictor, but the predictor can access both the explanation and the input example; (ii) there are no separate explainer and predictor at all — everything is produced jointly.

Approaches in (i) suffer from a similar faithfulness challenge as predict-and-explain methods do, because the predictor can make its decision *only* based on the input using whatever reasoning mechanism, while entirely ignoring the explanation Rajani et al. (2019); Kumar and Talukdar (2020).

For works of type (ii), given the input example as the prompt, a generation model outputs a continuation including both the explanation and the prediction in some designated order. This is analogous to any other generation task. Existing work along this line differs in the choice of the generation model and the end task. Early studies train task-specific models to generate free-text explanations for tasks like algebraic reasoning, NLI, sentiment classification, and QA. Ling et al. (2017); Narang et al. (2020). Tafjord et al. (2021); Dalvi et al. (2021); Hong et al. (2022) then develop methods to generate structured proofs for deductive reasoning, where each step is produced and verified iteratively to ensure faithfulness. All these methods require human-provided explanations as training data, while recent advances in the in-context learning ability of LLMs have shown promise in generating explanations in a few-shot fashion. These methods, which we have detailed in Section 2.3, prompt models to generate reasoning chains before the final answers, which prove effective in improving their performance on a variety of complex reasoning tasks (Wei et al., 2022b; Zhou et al., 2022).

3.5.3. Strengths and Weaknesses

In summary, self-explanatory models have several strengths. First, by definition, self-explanatory models provide built-in explanations, so there is **no need for post-hoc explanations**. Second, **the form of explanation is flexible**, e.g., model architecture, input features, natural language, or causal graphs. Third, it is possible to **supervise the explainer** with human-provided explanations.

This is helpful for learning more plausible explanations and encouraging the model to rely on desired human-like reasoning mechanisms instead of spurious cues. Finally, certain self-explanatory models (e.g., Tafjord et al., 2021; Hong et al., 2022; Chen et al., 2022b; Gao et al., 2022), are **faithful by construction** (we should be extra cautious about this claim, though).

Self-explanatory models also present a few key weaknesses. First, many such models cannot guarantee **faithfulness**, e.g., Neural Module Networks without intermediate supervision, predict-then-explain models, rationale-based explain-then-predict models, and certain jointly-predict-and-explain models. Second, the influence of explanations on **task performance** is mixed in self-explanatory models. Many studies discover a trade-off between performance and interpretability (Narang et al., 2020; Subramanian et al., 2020; Hase et al., 2020, i.a.), while others observe a positive impact on the performance from including explanations (e.g., CoT-style prompting). The effect highly depends on the task, the model family and size, the format of explanations, whether they are tuned, and how they are used (as inputs, targets, or priors) (Hase and Bansal, 2022; Lampinen et al., 2022a). Finally, large-scale human supervision on explanations can be **costly and noisy** (Dalvi et al., 2021). Also, it is **hard to automatically evaluate** the quality of model-generated explanations given reference human explanations, since there can be multiple ways to explain a prediction.

3.6. Case Study: CoT-style Explanations

It is claimed in the original CoT paper that LLM-generated explanations “provide an interpretable window into their behavior” (Wei et al., 2022b), leaving the impression that CoTs make models “self-explanatory”.

Using the properties outlined in Table 3.1, CoT-style explanations are produced at the same time as the prediction (*built-in*) with *white-box* access to the model at the time of generation, providing *local* (instance-level) explanations. Their unit of explanation is typically reasoning steps, and the form of explanation could be natural language, structured programs, or a combination of both. Therefore, CoT-style explanations can be considered as one instance of the last family, “self-explanations”, in our taxonomy. More specifically, they belong to the “jointly-predict-and-explain” subtype, as there are no separate explainer and predictor, but only one LLM that generates both the explanation and

prediction.

By comparing CoT-style explanations with existing methods of this family, we are able to develop a deeper understanding of why they lack faithfulness. Unlike previous studies in the jointly-predict-and-explain subtype, which iteratively generate and verify each step to form structured proofs (Tafjord et al., 2021; Dalvi et al., 2021; Hong et al., 2022), many CoT-style explanations generate all steps at once, without any verification on their faithfulness. Compared to explain-and-predict approaches, where the predictor can only access the explanation but not the input, CoT-style method has no such information bottleneck. The model can always access both the input and the explanation, which makes it technically possible for it to make the final prediction only based on the input, ignoring the explanation.

These inherent limitations can lead to CoT explanations that, while seemingly plausible and detailed, do not accurately reflect the model’s true reasoning process. Therefore, while CoT-style explanations offer a step towards more interpretable AI, they also highlight the ongoing challenges in achieving truly faithful explanations.

3.7. Summary

In this chapter, we have provided a comprehensive taxonomy of model explanation methods in NLP, categorizing them into five primary families: similarity-based methods, analysis of model-internal structures, backpropagation-based methods, counterfactual intervention, and self-explanatory models. We have examined the strengths and weaknesses of each category, with a particular focus on faithfulness.

Our case study on CoT-style explanations illustrated the potential and limitations of using LLMs to generate explanations. While CoT-style explanations represent a significant advancement in eliciting LLMs’ reasoning capabilities, they also reveal the challenges in ensuring these explanations are truly faithful to the model’s internal reasoning processes.

Building on the insights gained from this taxonomy, in the next chapter, we will introduce and evaluate a novel method aimed at enhancing the faithfulness of self-explanations generated by LLMs.

CHAPTER 4

Faithful Explanations with Symbolic Reasoning

This chapter discusses our paper titled “Faithful Chain-of-Thought Reasoning” (Lyu et al., 2023b), in which my collaborators Shreya Havaldar, Adam Stein, and I are co-first authors.¹²

As introduced in Section 2.3, Chain-of-Thought or CoT prompting (Wei et al., 2022b) is an in-context learning technique, where an LM generates a reasoning chain in free-text along with the final answer, given only a few exemplars. Compared to standard prompting (Brown et al., 2020a) where the LM only generates the final answer, CoT prompting has brought striking performance gains on many complex reasoning tasks, such as commonsense reasoning and math reasoning, where LMs have previously struggled.

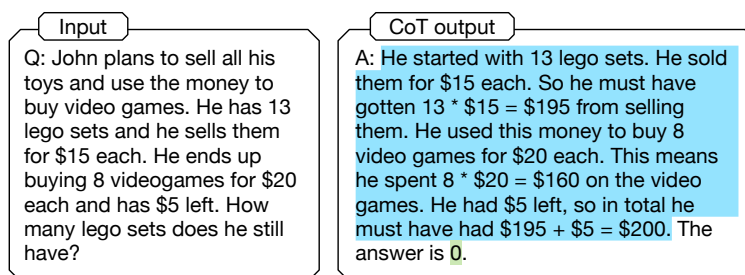


Figure 4.1: An example of *unfaithful* output from CoT prompting on GSM8K. The answer (green) does not follow from the reasoning chain (blue).

In addition to performance improvement, CoT is also claimed to “provide an interpretable window into the behavior of the model” (Wei et al., 2022b). However, as pointed out in Section 2.3, it lacks *faithfulness*. In the context of CoT, faithfulness means that the *reasoning chain* (explanation) should accurately represent *how the model arrives at the final answer* (its reasoning mechanism). In CoT-style prompting, the final answer does not necessarily follow from the previously generated reasoning chain, since the final answer is still produced via auto-regressive generation. Thus, there is no guarantee on faithfulness. Figure 4.1 exemplifies such an unfaithful generation from our inspection

¹²Concretely, I led the project, and worked on experiments on three domains as well as most of the analysis experiments; meanwhile, Shreya and Adam together worked on experiments on two domains and the human evaluation. In addition, we all contributed equally to the paper writing. I am using this work in my thesis with the permission of my co-authors and advisors.

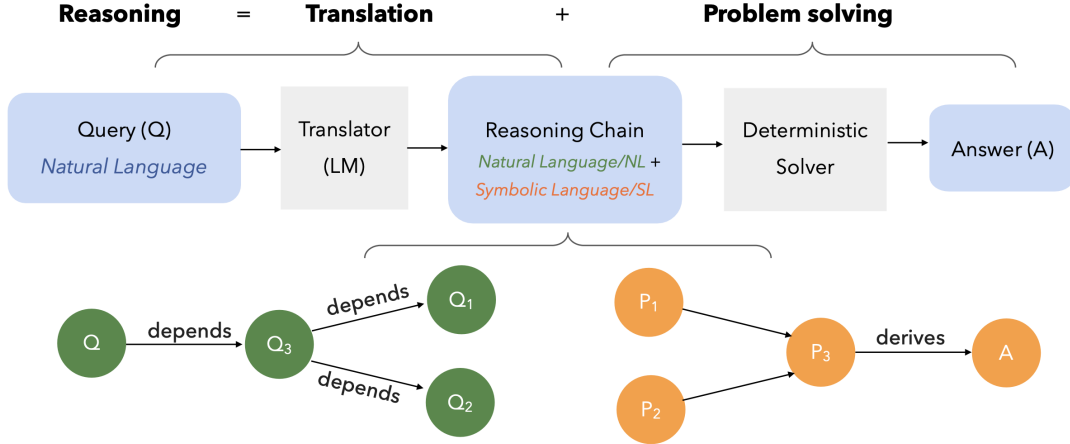


Figure 4.2: An overview of our 2-stage pipeline, consisting of **Translation**, where an LM translates a query (in NL/Natural Language) into a reasoning chain (which interleaves an NL decomposition of the original query into multiple inter-dependent subproblems, and SL/Symbolic Language programs to solve each subproblem); and **Problem Solving**, where an external solver executes the reasoning chain to derive the answer, thus ensuring faithfulness.

of the model output from (Wei et al., 2022b) on GSM8K: the answer “0” is not even mentioned in the reasoning chain. This illustrates that existing CoT methods do not provide true interpretability of how the model predicts the answer.

To bridge this gap, we propose **Faithful CoT**, a faithful-by-construction framework integrating symbolic reasoning with stochastic generation. Specifically, we break down a complex reasoning task into two sequential stages: **Translation** and **Problem Solving** (Figure 4.2). During Translation, an LM translates a Natural Language (NL) query into a reasoning chain, which interleaves NL and a task-dependent Symbolic Language (SL), such as Python, Datalog, or Planning Domain Definition Language (PDDL). The NL part decomposes the original query into a dependency graph of multiple simpler subproblems, and the SL part solves each subproblem individually. Then, in the Problem Solving stage, the reasoning chain is executed by a deterministic solver, e.g., a Python/Datalog interpreter, or a PDDL planner, to derive the answer. This process guarantees that the reasoning chain has to be a faithful explanation of how the answer is derived, since the answer is the result of deterministically executing the chain.

We evaluate our approach on 10 reasoning datasets from 4 diverse domains: Math Word Problems

(MWP), Planning, Multi-hop QA, and Relational Inference. We compare it with standard prompting (Brown et al., 2020a) and CoT prompting, using the same underlying LM (Codex (Chen et al., 2021)) and the same decoding strategies (greedy and self-consistent decoding (Wang et al., 2022b)). Results show that on 8 out of the 10 datasets, our approach outperforms both baselines, with an average accuracy gain over CoT of 4.4 on MWP, 1.9 on Planning, 4.0 on Multi-hop QA, and 18.1 on Relational Inference, using greedy decoding. Together with self-consistent decoding, we achieve new state-of-the-art few-shot performance at the time of publication on 7 out of the 10 datasets.

4.1. Method

Our method, **Faithful CoT**, is a 2-stage pipeline. Given a complex question Q , we prompt an LLM to generate a reasoning chain C , which is then used to derive the final answer A . Like previous CoT-style work, our prompt consists of (Q, C, A) triples. Notable differences lie in our unique interleaving of NL (natural language) and SL (symbolic language) in C , as well as the way we derive the final answer A .

In the **Translation** stage, given a complex query Q in NL, we prompt an LM to translate it into a reasoning chain C , which interleaves NL and a task-specific SL (e.g., Python, Datalog, or PDDL).¹³ In the **Problem Solving** stage, we call a deterministic external solver, e.g., a Python interpreter, a Datalog executor, or PDDL planner, depending on the task, to obtain the answer A from the reasoning chain C . We define C_{NL} to be the NL component and C_{SL} to be the SL component in C . Though we separate the two components notationally, they are interleaved in the generation. Using this approach, C is guaranteed to be a faithful model explanation, since our final A is the result of deterministically executing C_{SL} . Moreover, C_{NL} allows the user to better understand the reasoning process.

We apply this method to 4 types of complex reasoning tasks: MWP, Multi-hop QA, Planning, and Relational Inference. Next, we will illustrate how our method works for each of them with examples.

¹³Our prompts can be found in the Supplementary Materials.

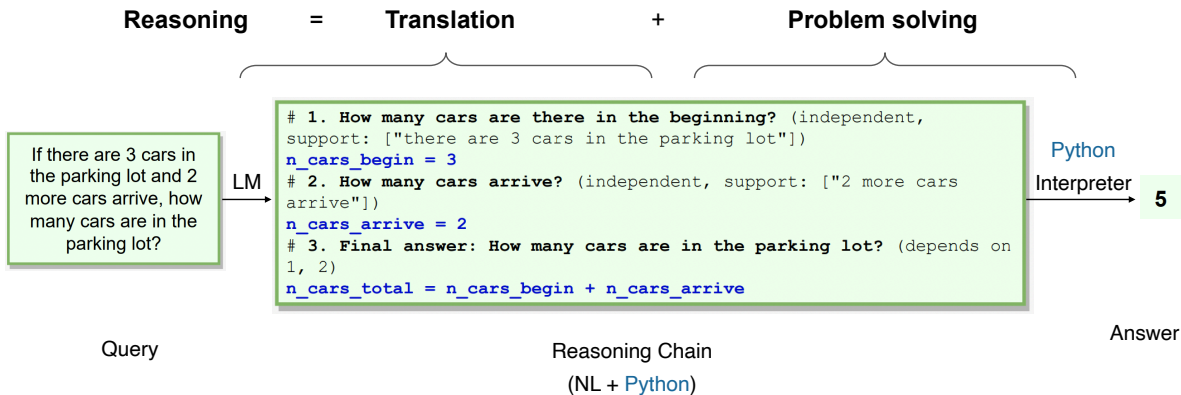


Figure 4.3: An example of how our method can be applied to Math Word Problems (input from the GSM8K dataset). “Independent”/“Depends on X” represents the dependency relationship between sub-problems. “Support” represents the rationale for how to solve each sub-problem.

4.1.1. Math Word Problems (MWP)

Given a grade-school math question Q written in NL (“If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?”, in Figure 4.3), we want to obtain A as a real-valued number (5). In the Translation stage, we prompt the LM to take in Q and generate a reasoning chain C , which interleaves C_{NL} and C_{SL} . Specifically, the C_{NL} component consists of three types of information:

- (a) **Subquestions:** Q is broken down into multiple smaller-scale subquestions, e.g., “1. how many cars are there in the beginning?”, “2. how many cars arrive?”, and “3. how many cars are in the parking lot?”.
- (b) **Dependency Graph:** As illustrated in Figure 4.2, each subquestion can either be answered directly via context (subquestions 1 and 2 are “independent”) or rely on answers to previous subquestions (subquestion 3 “depends on 1 and 2”).
- (c) **Rationales:** Each subquestion is accompanied with rationale(s) to support the answer (the “support” field). The rationales can be either a subset of the original context (“2 more cars arrive”) or any external knowledge (“there are 7 days in a week”) relevant to the subquestion.

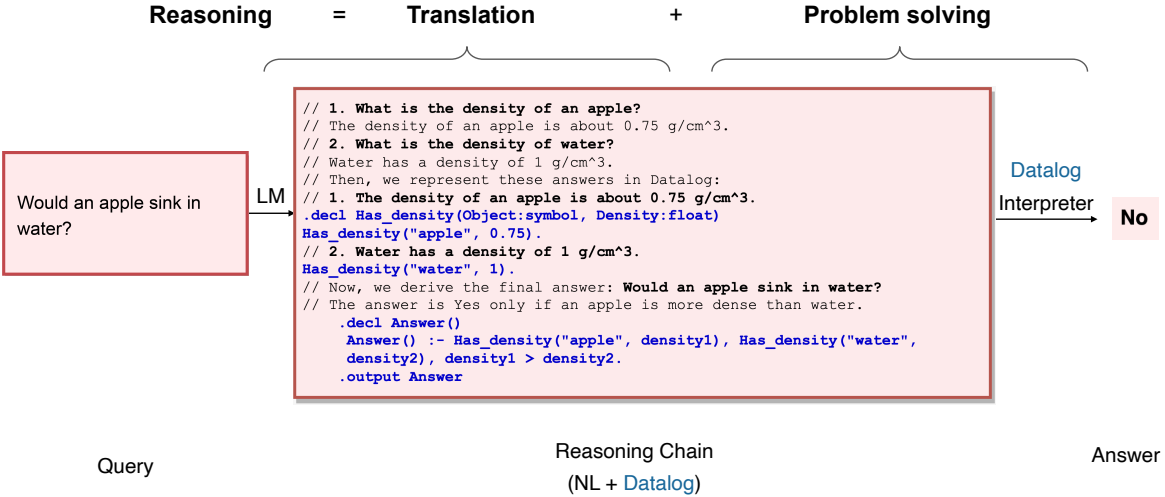


Figure 4.4: An example of how our method can be applied to Multi-hop QA (input from the StrategyQA dataset).

Each subquestion and its corresponding dependencies and rationales inform the subsequent generation of C_{SL} . In our example in Figure 4.3, C_{SL} consists of Python code generated to answer each subquestion in C_{NL} . During the Problem Solving stage, we execute C_{SL} using our solver, a Python interpreter, to derive A (5 cars in the end).

4.1.2. Multi-hop QA

Given a complex question Q that involves multiple steps of reasoning (e.g., “Would a pear sink in water?” in Figure 4.4), we want to obtain the answer A as a Boolean value or string value variable. Similar to our MWP task formulation, C interleaves C_{NL} (NL comments), and C_{SL} (symbolic program). Depending on the nature of the task, the format of the reasoning chain C is slightly different: for some datasets (e.g. StrategyQA), the LM first generates all subquestions and their answers in NL, and then represents these answers as SL to derive A ; for others (e.g. Date Understanding and Sports Understanding), the LM interleaves the NL subquestions and the SL program, similar to the case of MWP. In terms of SL, we use both Python and Datalog¹⁴, also depending on the dataset. As Multi-hop QA problems involve multi-step reasoning to solve, C_{SL}

¹⁴Datalog is a declarative logic programming language, often used as a query language for deductive databases. According to <https://en.wikipedia.org/wiki/Datalog>, “A Datalog program consists of facts, which are statements that are held to be true, and rules, which say how to deduce new facts from known facts.”

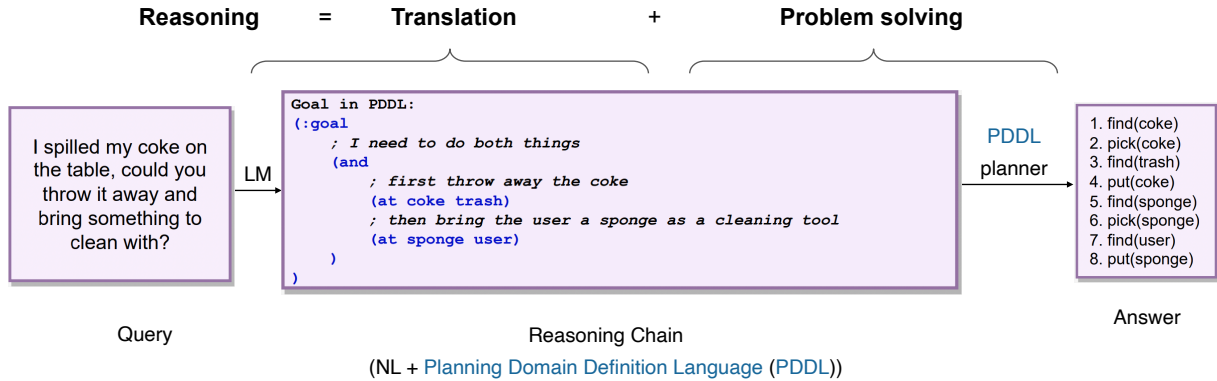


Figure 4.5: An example of how our method can be applied to Planning (input from the SayCan dataset).

often utilizes Boolean algebra and string comparisons (in Python) along with relation definitions and logic programming (in Datalog). We use their corresponding interpreter as our deterministic solver to execute C_{SL} and obtain A .

In the example from Figure 4.4, the LM first generates the subquestions, “1. What is the density of a pear?” and “2. What is the density of water?”, which are individually answered in NL. The answers (“Water has a density of $1g/cm^3$ ”) are converted to Datalog statements ($Has_density(“water”, 1)$), which are then combined to formalize the truth condition of the final answer. Finally, we execute the Datalog program to determine that a pear would **not** sink in water.

4.1.3. Planning

In a user-robot interaction scenario, given a household task query Q from a user, we want to come up with a plan of actions A that the robot should take in order to accomplish the task. For example, in Figure 4.5, given user query “I spilled my coke on the table, could you throw it away and bring something to clean with?”, a possible plan can be “find(coke), pick(coke), find(trash), put(coke) ...”. In the Translation stage, an LM translates Q into C , consisting of C_{NL} (which breaks down Q into subtasks) and C_{SL} (which represents the subtasks as a symbolic goal in PDDL¹⁵ Figure 4.5

¹⁵PDDL is a language to define and solve classical planning problems. A goal is a special construct in PDDL. According to https://en.wikipedia.org/wiki/Planning_Domain_Definition_Language, “PDDL is a human-readable format for problems in automated planning that gives a description of the possible states of the world, a description of the set of possible actions, a specific initial state of the world, and a specific set of desired goals.”

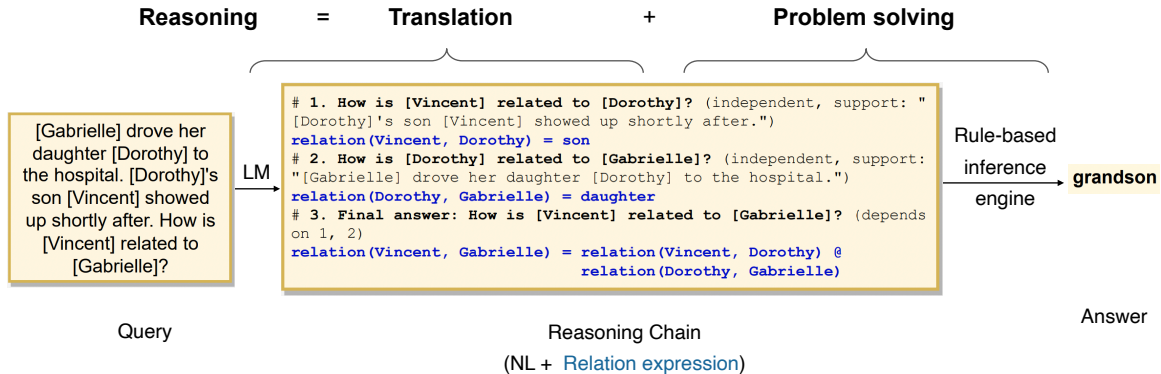


Figure 4.6: An example of how our method can be applied to Relational Reasoning (input from the CLUTRR dataset).

shows this translation, with C_{SL} in blue and C_{NL} in black. Finally, we call a PDDL Planner as the deterministic solver to obtain A , a plan to accomplish the goal C_{SL} under the predefined scenario.

4.1.4. Relational Inference

Given a relational inference problem Q written in NL, we want to obtain A as a string-valued variable. For example, the **CLUTRR** (Compositional Language Understanding and Text-based Relational Reasoning) dataset (Sinha et al., 2019) involves inferring the family relationship (e.g., “grandson”) between two people from a short story (e.g., “[Gabrielle] drove her daughter [Dorothy] to the hospital. [Dorothy]’s son [Vincent] showed up shortly after. How is [Vincent] related to [Gabrielle]?”, shown in Figure 4.6). During the Translation stage, we prompt the LM to generate C , consisting of C_{NL} and C_{SL} . Similar to previous tasks, C_{NL} breaks down Q into subquestions (“How is [Vincent] related to [Dorothy]” and “How is [Dorothy] related to [Gabrielle]”), as well as provide input extracts as rationales to support the answer (“[Dorothy]’s son [Vincent] showed up shortly after”, etc.). Each subquestion in C_{NL} is answered in C_{SL} via a custom relation expression representing the relation between the mentioned entities, for example, `relation(Vincent, Dorothy)=son` denotes that Vincent is Dorothy’s son. In the Problem Solving stage, our solver is a simple relation inference function that relies on a set of transitivity rules provided by (Zhang et al., 2022a) among possible family relationships, e.g., `son@daughter=grandson` (the son of one’s daughter is one’s grandson). Our solver recursively applies these rules on C_{SL} to derive A , and determine that Vincent is Gabrielle’s

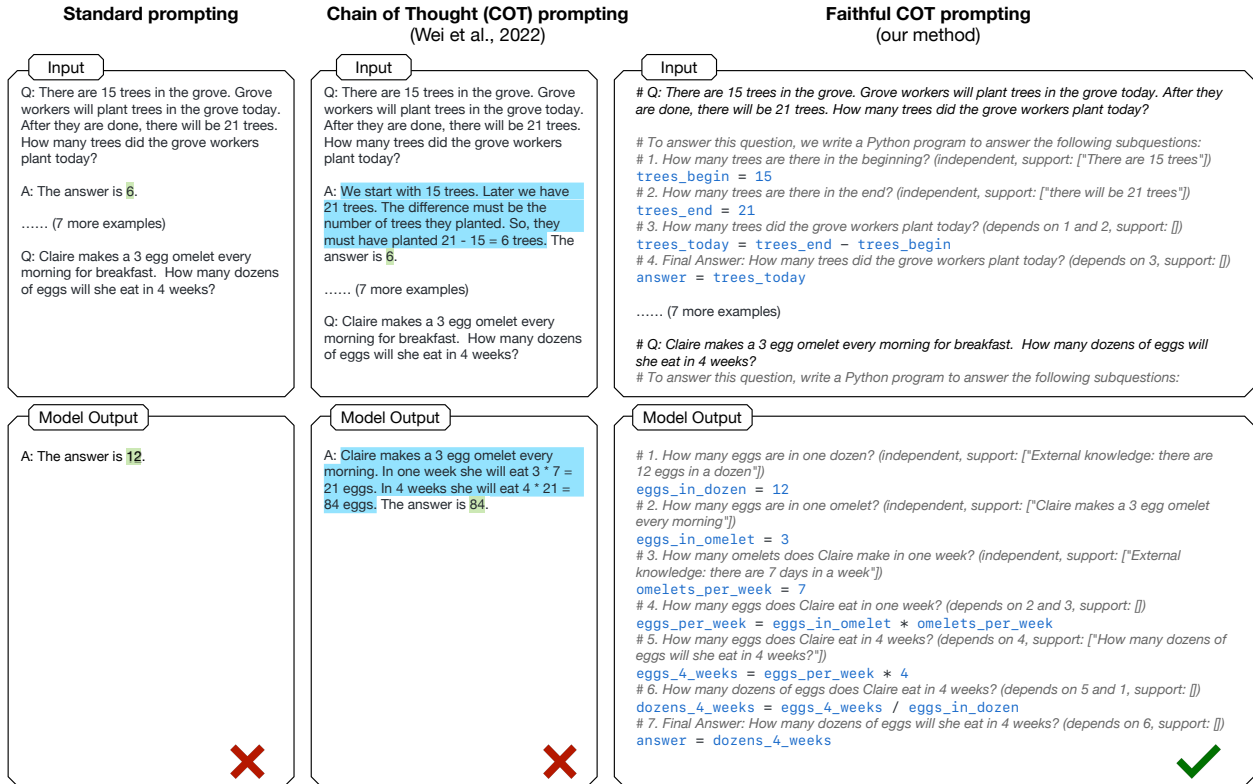


Figure 4.7: The prompt for GSM8K (a Math Word Problems dataset) from standard, CoT, and Faithful CoT prompting (ours). The ground-truth answer is 7, and only our method correctly computes the answer.

grandson.

4.2. Experimental setup

4.2.1. Datasets

Here, we summarize the evaluation datasets used for each domain. We select the same number (6 to 10, depending on the task) of exemplars as in (Wei et al., 2022b) to form our few-shot prompt.¹⁶ Unless otherwise stated, we use the official splits: training set for exemplar selection, validation set for prompt tuning, and test set for evaluation.¹⁷

Math Word Problems (MWP): We follow (Wei et al., 2022b) and consider the same five MWP benchmarks: **GSM8K** (Cobbe et al., 2021), **SVAMP** (Patel et al., 2021), **MultiArith**

¹⁶Prompt exemplars can be found in Appendix A.4.

¹⁷Dataset Details can be found in Appendix A.2.

(Roy and Roth, 2015), **ASDiv** (Miao et al., 2020), and **AQuA** (Ling et al., 2017). For all datasets, the input question is phrased in NL. The answer is a string-valued mathematical expression for AQuA, and one or more integer(s) for all other datasets. We use the same 8-shot prompt for all datasets except AQuA.

Multi-hop QA: We consider the three datasets: **StrategyQA** (Geva et al., 2021), a dataset of open-domain questions that require an implicit multi-step strategy to answer, e.g., “Did Aristotle use a laptop?” involves answering “1. When did Aristotle live?”, “2. When was the laptop invented?”, and “3. Is #2 before #1?”; **Date Understanding** from BIG-bench (BIG-Bench collaboration, 2021), which asks the model to infer a date from a context, by performing computation on relative periods of time; and finally, **Sports Understanding** from BIG-bench, which asks the model to decide whether an artificially constructed statement related to sports is plausible or implausible. Since the latter two datasets do not have a training set, we follow (Wei et al., 2022b) and select 10 examples from the test set to form the prompt and use the rest for evaluation.

Planning: We use the **SayCan** dataset (Ahn et al., 2022), which assumes a scenario of a robot operating in a kitchen, helping the user with household tasks, e.g., “bring a coke to the table”. There are a number of locations and objects that the robot can interact with. The robot can only perform a fixed set of actions, including `find`, `pick`, and `put`. The task is to map a user query in NL to a plan of predefined actions. Following (Wei et al., 2022b), we manually write 7 exemplars, since no training set is provided.

Relational inference: We use the **CLUTRR** (Sinha et al., 2019) benchmark described above. The dataset has multiple splits based on the number of intermediate steps K required to reach the answer. We construct the prompt using 8 exemplars with $K \in \{2, 3\}$, and test the models on the remaining examples with K up to 10.

4.2.2. Evaluation Metrics

We evaluate the model performance with the accuracy of the final answer. Following previous work (Wei et al., 2022b; Wang et al., 2022b; Chen et al., 2022b), for all MWP datasets (except AQuA)

where the answer contains integer(s), a correct answer is defined as the exact match between the prediction and the ground truth both rounded up to the nearest integer (with the `math.ceil()` function in Python); for StrategyQA and Sports Understanding where the answer is a Boolean value, it is defined as the exact match between the prediction and the ground truth both evaluated as a Boolean variable; for SayCan, the generated plan is considered correct if it is among the ground truth plans; for all other datasets, we rely on the exact match between the prediction string and the ground truth string.

4.2.3. Language Model

In our main experiments, we use OpenAI Codex (Chen et al., 2021) (`code-davinci-002`, with 175B parameters) as the underlying LM during Translation.

4.2.4. Baselines

We compare our method to two other baselines, shown in Figure 4.7: **standard** few-shot prompting, popularized by (Brown et al., 2020a), with demonstrations of only the question and the answer (green); **CoT** prompting (Wei et al., 2022b), which additionally provides a reasoning chain in NL (blue); and **Least-to-Most** prompting, which also prompts the model to decompose the original question into a set of subquestions and solves them sequentially, but does not involve dependency relation or symbolic structure. We also show the published SOTA few-shot results at the time of publication of this work.

All prompting methods are compared under two decoding strategies: **greedy** decoding, where the LM samples the most probable next token from the vocabulary (i.e., temperature = 0.0); and **self-consistency** decoding (Wang et al., 2022b), where the LM generates multiple reasoning chains and chooses the final chain based on majority voting on the evaluated answer (we use a temperature of 0.4 and 40 generations for all datasets).¹⁸ We compare greedy and self-consistency sampling as they are two widely used decoding strategies for LMs, and self-consistency has been shown to consistently improve LMs’ performance on multiple reasoning tasks (Wang et al., 2022b). We

¹⁸Note that we do not report the performance of standard prompting with self-consistency decoding, since when the number of sampled outputs is large enough, this converges to standard prompting with greedy decoding (Wang et al., 2022b).

Method	Math Word Problems				Planning	Multi-hop QA			Relation	
	GSM8K	SVAMP	MultiArith	ASDiv	AQuA	SayCan	StrategyQA	Date	Sport	CLUTRR
Greedy Decoding										
Standard	19.6	69.5	43.8	72.1	31.5	82.5	63.9	51.3	71.9	42.0
CoT	63.3	77.3	96.5	80.0	42.1	86.4	72.5	59.9	98.6	48.5
Least-to-Most	38.3	80.3	74.0	76.5	40.6	77.7	72.2	76.6	99.5	47.2
Faithful CoT (ours)	72.3	83.4	98.8	80.2	47.2	89.3	63.0	81.6	99.1	58.9
Self-Consistency Decoding										
CoT	78.0	86.8	100.0	84.2	52.0	89.3	79.8	63.8	98.0	45.7
Least-to-Most	38.8	80.5	74.0	76.3	44.9	76.7	71.9	77.2	99.4	50.9
Faithful CoT (ours)	80.0	88.8	99.2	84.4	61.4	94.2	65.2	85.5	99.0	71.9

Table 4.1: Accuracy of different methods on 10 reasoning datasets from 4 domains. We compare our method, Faithful CoT, with standard Brown et al. (2020a), CoT Wei et al. (2022b), and Least-to-Most prompting Zhou et al. (2022), with `code-davinci-002` as the LM. The best results within each decoding strategy are **bolded**.

reproduce the baseline results ourselves in cases when they are not reported or when we clean the test set.

4.3. Results

Our results on all datasets are shown in Table 4.1. We see that Faithful CoT outperforms CoT across most datasets and domains for both greedy and self-consistency decoding.

With greedy decoding, our method outperforms CoT on 9 of the 10 benchmark datasets spanning the 4 domains. On average, it improves over CoT in the MWP domain by 4.4, in Planning by 1.9, in Multi-hop QA by 4.0, and in Relational Inference by a surprising 18.1.

Our method also outperforms CoT under self-consistency decoding on 8 out of the 10 datasets. Compared to greedy decoding, the average accuracy gain becomes larger for Planning (1.9 \rightarrow 3.9) and Relational Inference (18.1 \rightarrow 26.2), but smaller for MWP (4.4 \rightarrow 2.5) and Multi-hop QA (4.0 \rightarrow 2.7). Also, it is worth noting that our method achieves the new few-shot SOTA results on 7 datasets at the time of publication.

4.4. Analysis

In this section, we analyze the role of different components in our pipeline, to better understand where its capabilities come from and where it still struggles. Unless otherwise stated, we choose one dataset from each domain for analysis – GSM8K, Date Understanding, SayCan, and CLUTRR –

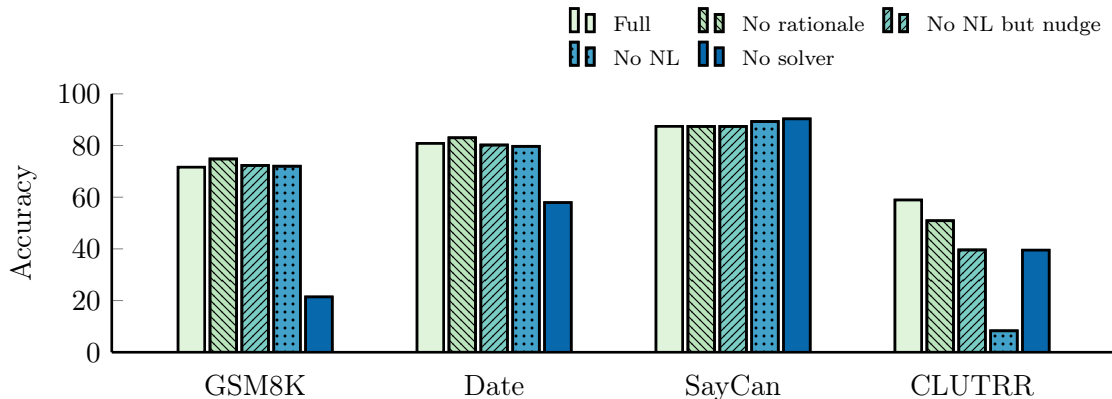


Figure 4.8: Ablation study results: accuracy when we remove different components of our Faithful CoT pipeline. See Section 4.4.1 for details.

using greedy decoding.

4.4.1. Ablation Study

Given the strong performance of Faithful CoT, we now address a natural question: **how much does each part of the prompt contribute to the accuracy?** We perform an ablation study where we remove different parts of the prompt and see how the performance changes. In addition to the original prompt (“Full”), we test four variations, illustrated with the example from Figure 4.7:

No rationale: We remove the rationales, i.e., everything in the brackets from the NL comments, e.g., “independent, support: [‘There are 15 trees’]”.

No NL but nudge: We remove all NL comments except the “nudge” line: e.g., “# To answer this question, we write a Python program to answer the following subquestions”.

No NL: We remove all NL comments.

No solver: Instead of calling the external solver, we add “Answer: {answer}” to the end of every exemplar and let the LM predict the answer itself.

Figure 4.8 shows the results of all prompt variations. On GSM8K, Date Understanding, and SayCan, NL comments contribute little to the performance, and sometimes even slightly hurt it. On CLUTRR, however, their role is crucial, since the exclusion of each component (rationale, nudge, subquestions)

Exemplars	GSM8K	Date	SayCan	CLUTRR
Set 0 (Table 4.1)	71.6	80.8	89.3	58.9
Set 1	72.3	81.3	90.3	59.0
Set 2	70.8	85.0	85.4	57.2
Set 3	71.6	82.5	88.3	58.0
Set 4	70.6	77.4	88.3	55.5
Set 5	68.5	85.0	89.3	56.0
Mean	70.9	82.0	88.5	57.4
Std.	1.3	2.9	1.7	1.5

Table 4.2: Robustness to the choice of exemplars across 6 runs.

results in a clear accuracy drop. In particular, comparing **No NL but nudge** and **No NL**, the nudge line itself brings a striking improvement by 31.3 points.

The external solver relieves the burden of problem solving from the LM. Without it, the accuracy suffers a huge decline on GSM8K, Date Understanding, and CLUTRR (-50.8, -22.9, and -19.4 respectively), while on SayCan it improves by 2.9 nonetheless. One potential influencing factor is that SayCan might be too homogeneous, as it contains a set of only 3 predefined actions. This can make the task relatively easy, which allows all model variants to achieve around 90% accuracy and renders the solver unnecessary. Another potential reason is the level of correspondence between the final answer and the reasoning chain for different datasets: as shown in Figure 4.5, the answer in SayCan is a sequence of actions (e.g., `find(redbull)`), each directly corresponding to one step in the reasoning chain (e.g., `at redbull trash`). However, the answer in the other three datasets is only a single number or string, which can only be derived after executing *all* the steps in the reasoning chain. Therefore, the latter type of tasks further necessitates the presence of an external solver.

4.4.2. Robustness to Exemplars

We now answer the question: **how much does the choice of exemplars matter?** To do this, we annotate 20 examples in total, randomly sample k (7-10, depending on the dataset) from our annotated examples to construct the prompt, and repeat the process five times. Table 4.2 shows the performance of all six runs, including the original (from Table 4.1). The mean accuracy is close to the original (-1.5 to +1.2), still above the baselines by a large margin (7 to 17) on all datasets except the arguably easiest SayCan, considering the standard deviation (1.3 to 2.9). This strongly suggests that the benefits of Faithful CoT are minimally influenced by the choice of exemplars.

4.4.3. Robustness to Prompt Phrasing

Prompt	GSM8K	Date	SayCan	CLUTRR
Original	72.3	81.6	89.3	58.9
Variation 1	69.1	84.4	88.3	-
Variation 2	70.3	81.6	90.3	56.2
Variation 3	70.2	80.5	87.4	55.9
Mean	70.5	82.0	88.8	57.0
Std	1.3	1.7	1.3	1.7

Table 4.3: Robustness to prompt phrasing.

We study the sensitivity of our method to subtle differences in the prompt design. We experiment with three prompt variations: 1. randomly permuting the order of independent subquestions/reasoning steps; 2. Changing the variable names; 3. changing the nudge line (e.g. from “# To answer this question, write a Python program to answer the following subquestions” to “# To solve this question, we answer each of the following subquestions with a Python program”).

We rerun the evaluation of all three variations under greedy decoding. Table 4.3 shows the results. Overall, the performance is quite stable, always above each baseline on all four datasets.

4.4.4. Model Sensitivity

Now, we want to answer the question: **how much does the choice of LM matter?** We examine the effect of using four alternative code-generation models as the Translator: `code-davinci-001`, `text-davinci-002`, `text-davinci-003`, `gpt-4`. We compare our method with the three baselines using each of the above LM on five MWP datasets, using the greedy decoding strategy.

As shown in Table 4.4, regardless of the underlying LM, Faithful CoT consistently outperforms all baselines on the vast majority of the datasets, and performs very closely with the best method (<2.0 difference) on the remaining ones. On average, it has a relative accuracy gain of 16.1%, 11.0%, 9.4%, and 4.6 % over the best-performing method among the baselines, for each LM respectively. This indicates that even though the absolute performance varies depending on the LM, Faithful CoT brings a relatively consistent accuracy gain.

Notably, with GPT-4 as the underlying LM, Faithful CoT results in 95.0+ accuracy in 4 of the 5 MWP datasets, far outperforming the previous few-shot SOTA on three of them (GSM8K, SVAMP,

and ASDiv).

Method / Dataset	GSM8K	SVAMP	MultiArith	ASDiv	AQuA	Average
LM: code-davinci-001						
Standard	5.2	28.7	8.6	38.5	22.8	20.8
CoT	14.7	41.2	57.2	50.4	22.4	37.2
LtM	9.8	44.9	18.5	46.6	20.5	28.1
Faithful CoT (ours)	27.4	50.8	63.3	53.7	20.9	43.2
LM: text-davinci-002						
Standard	15.4	65.2	34.1	64.8	24.0	40.7
CoT	47.2	68.0	91.1	69.9	40.6	63.4
LtM	32.9	73.8	68.3	70.9	34.6	56.1
Faithful CoT (ours)	62.7	80.0	92.8	75.4	41.3	70.4
LM: text-davinci-003						
Standard	16.9	69.4	38.8	59.1	29.5	42.7
CoT	59.6	79.5	95.0	69.1	46.9	70.0
LtM	34.6	79.5	73.7	70.0	44.5	60.5
Faithful CoT (ours)	71.7	85.1	94.5	80.7	50.8	76.6
LM: gpt-4						
Standard	46.9	88.4	98.7	70.2	50.4	70.9
CoT	64.9	80.0	94.0	71.6	75.2	77.1
LtM	91.8	92.9	98.3	86.7	72.0	87.5
Faithful CoT (ours)	95.0	95.4	98.5	95.6	73.6	91.6

Table 4.4: Accuracy of different prompting methods with each underlying LM on 5 MWP reasoning datasets.

4.4.5. Error Analysis

To further investigate where our method still fails, we inspect 100 errors¹⁹ from model predictions on each of the four datasets and manually annotate the error categories. We only present the results on GSM8K here, and leave the error analyses on other datasets in Appendix A.3.1.

As shown in Figure 4.9, we categorize the errors on GSM8K into 6 types, inversely sorted with frequency:

Wrong Subquestion (49%): The LM produces a wrong NL subquestion, which eventually leads to the incorrect answer. While this is the majority error type in our sample, it is worth noting that in a typical human-in-the-loop collaboration, these errors are easily fixable. Even if the user is unfamiliar with programming, they can inspect the NL subquestions and potentially correct the model error by simply deleting or editing a wrong subquestion.

¹⁹To encourage sample diversity, we embed all the errors using text-davinci-002 and cluster the embeddings using spectral clustering. This produces around 70 clusters of different sizes, from which we gather 100 samples using importance sampling.

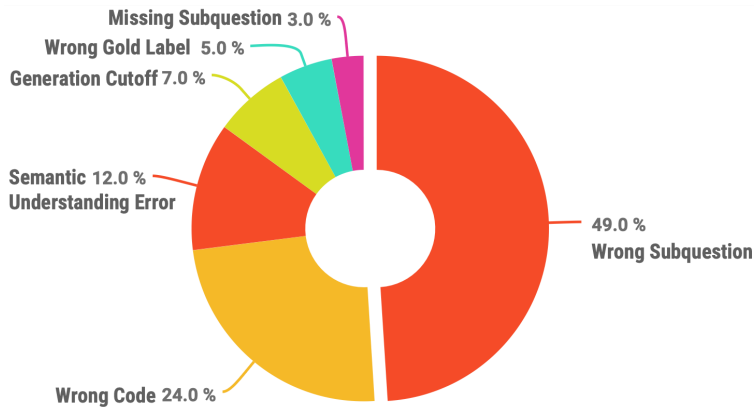


Figure 4.9: Error analysis for GSM8K. For a detailed description of the error categories, see Section 4.4.

Wrong Code (24%): The NL subquestion is correct, but the code fails to answer the subquestion correctly. For example, the code uses a variable that has not been previously defined. Note that this type of error can be reduced as LMs improve their code generation ability, for example, by continued finetuning on the relevant SL.

Semantic Understanding Error (12%): The LM incorrectly interprets certain semantic subtleties in the query. This is the most complex and most interesting error category. For example, consider the following problem:

*If Martin eats Cheerios every day for breakfast, he'll **lose** 1.25 pounds/week. If he eats donuts every day for breakfast, he'll **gain** 1.75 pounds/week. What will be the difference in his weight at the end of 5 weeks between the two breakfast options?*

As shown in Table 4.5, the generated code does not assign opposite polarities (signs) for “pounds lost” vs. “pounds gained”. For other examples in this category, we notice errors like missing that a pair of something has 2 items in it, missing to subtract 2 for “two years ago” when it occurs as a subjunctive, and so on. Fixing these errors, in general, will require more than providing additional examples in the prompt.

Generation Cutoff (7%): The generation stops midway, mainly due to the LM producing the same steps over and over again. These errors could be easily detected in postprocessing and possibly fixed by re-prompting the LM.

```

# 1. How many pounds will Martin lose per week if he eats Cheerios every day for breakfast? (independent,
support: ["he'll lose 1.25 pounds/week"])
pounds_lost_cheerios = 1.25
# 2. How many pounds will Martin gain per week if he eats donuts every day for breakfast? (independent,
support: ["he'll gain 1.75 pounds/week"]) pounds_gained_donuts = 1.75
# 3. How many weeks are there in 5 weeks? (independent, support: ["External knowledge: there are 7 days
in a week"])
weeks_in_5_weeks = 5
# 4. How many pounds will Martin lose in 5 weeks if he eats Cheerios every day for breakfast? (depends on
1 and 3, support: [])
pounds_lost_cheerios_5_weeks = pounds_lost_cheerios * weeks_in_5_weeks
# 5. How many pounds will Martin gain in 5 weeks if he eats donuts every day for breakfast? (depends on 2
and 3, support: [])
pounds_gained_donuts_5_weeks = pounds_gained_donuts * weeks_in_5_weeks
# 6. What will be the difference in his weight at the end of 5 weeks between the two breakfast options?
(depends on 4 and 5, support: [])
difference_5_weeks = pounds_gained_donuts_5_weeks - pounds_lost_cheerios_5_weeks
# 7. Final Answer: What will be the difference in his weight at the end of 5 weeks between the two breakfast
options? (depends on 6, support: [])
answer = difference_5_weeks

```

Table 4.5: Generated code for the MWP problem in Section 4.4 – Error Analysis.

Wrong Gold Label (5%): We find 5 (out of our 100) examples that are genuine annotation errors in the gold labels.

Missing Subquestion (3%): The LM misses a relevant subquestion needed for the rest of the reasoning chain to work. These errors are also potentially fixable via human-in-the-loop interaction, where the user can insert a subquestion into the reasoning chain.

4.5. Conclusion

In this chapter, we focused on the faithfulness of CoT-style “self-explanations” – to what extent they accurately capture the reasoning mechanism used by LLMs in predicting the answer. Due to their free-text form, reasoning chains generated by conventional CoT prompting are not necessarily faithful explanations. To address this gap, we introduce the Faithful CoT framework, an innovative approach that integrates stochastic translation with deterministic problem-solving. This method does not merely provide more faithful explanations, but also empirically enhances the performance of the end task. Through a detailed analysis, we show that the external solver is essential to both explanation faithfulness and task accuracy.

Our findings shed light on the interplay between interpretability and performance. A number

of previous studies in self-explanatory models have shown an interpretability-performance trade-off (Camburu et al., 2018b; Kumar and Talukdar, 2020, i.a.), however, by incorporating symbolic reasoning into end-to-end NLP workflows, we demonstrate that there can exist a synergy between them. This suggests a potentially promising pathway to enhance interpretability without sacrificing performance in self-explanatory models.

One important limitation of our work lies in the range of tasks. Currently, we have only focused on tasks that can be formulated as a symbolic language. For tasks that intuitively cannot be solved with symbolic reasoning, such as commonsense QA, Natural Language Inference (NLI), and poem writing, will it be possible to adapt our framework to them? Another limitation concerns the scope of faithfulness. We now only guarantee that the problem solving stage is faithful, but the translation stage is still opaque, meaning it is not self-interpretable how the LM generates the reasoning chain from the question. It is still an under-explored question whether it is possible to improve the interpretability of the LM generation process in general, and a few recent studies have made promising early progress (Yin and Neubig, 2022; Sarti et al., 2023) that can be useful in improving the faithfulness of the Translation stage.

Even as we address the issue of faithfulness, a crucial question remains: does faithfulness translate into practical utility? For instance, can faithful explanations make models more trustworthy and robust in real-world applications? This question is central to the next phase of our exploration. In the following chapters, we will investigate whether these explanations can be leveraged to enhance the practical utility of LMs. Specifically, we will explore how explanations can improve model trustworthiness and robustness, examining scenarios such as calibrating prediction confidence, correcting prediction errors, and mitigating reliance on spurious cues.

CHAPTER 5

Using Explanations for Confidence Calibration

This chapter discusses our paper titled “Calibrating Large Language Models with Sample Consistency” (Lyu et al., 2024), in which my collaborators Kumar Shridhar and I are co-first authors.²⁰

In this chapter, we will study whether explanations can be used to make LMs better *calibrated*, meaning that the confidence level is aligned with the likelihood of the prediction being correct (Brier, 1950). A well-calibrated system can enable model developers to provide selective predictions, help users decide when to trust or distrust model responses, and potentially facilitate performance improvement through human intervention or self-refinement (Madaan et al., 2023; Shridhar et al., 2023).

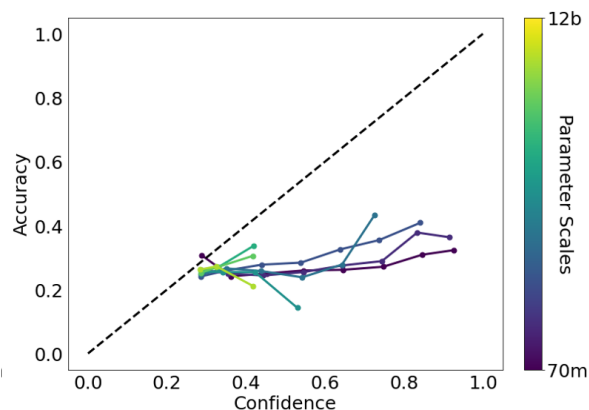


Figure 5.1: Calibration curves of LLMs of different sizes on MMLU, a widely-used benchmark for LM’s reasoning abilities, where the x-axis is the probability of the prediction and the y-axis is the accuracy (figure from Zhu et al. (2023)). Different colors of lines represent LLMs of different sizes, as shown on the parameter scale. The dotted line represents “perfect calibration”, where the accuracy is exactly aligned with confidence.

Unfortunately, LLMs are not well-calibrated off-the-shelf — as shown in Figure 5.1, the probability scores of model predictions are often poorly aligned with actual performance (Jiang et al., 2021;

²⁰We came up with the core idea together in a meeting. Then, I led the project, and worked on experiments for closed-source LLMs; meanwhile, Shridhar worked on experiments for open-source LLMs. In addition, we both contributed equally to paper writing. I am using this work in my thesis with the permission of my co-authors and advisors.

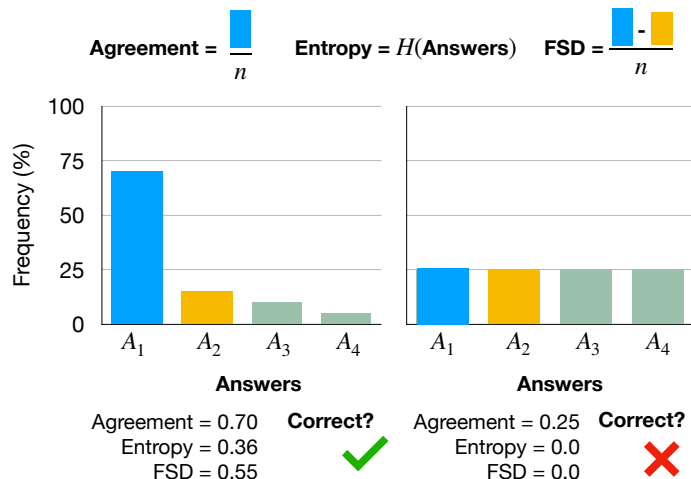


Figure 5.2: We study three consistency measures in this work: agreement-based, entropy-based, and first-second-distance-based (FSD). Does a more “consistent” distribution (left) indicate a higher confidence in correctly answering the question than a less “consistent” distribution (right)?

Chen et al., 2023b). While traditional calibration methods (Guo et al., 2017; Lakshminarayanan et al., 2017; Gal and Ghahramani, 2016, i.a.) can in theory be used to better calibrate open-source LMs, for recent LLMs, these methods sometimes become formidably costly because of the need to retrain multiple copies of the model, and might even be inapplicable due to inaccessible training data, model weights, and output probabilities in closed-source LLMs.

In this work, we investigate the research question: *Can explanations help make models better calibrated?* One potential way is through sampling multiple **explanations** given a single query (Wang et al., 2022b) and measuring the *consistency*²¹ of the distribution. Intuitively, “spiked” distributions (Figure 5.2, left) could indicate that the model is confident in correctly answering the question, whereas close to uniform distributions (Figure 5.2, right) might be correlated with lower confidence. If these intuitions hold, this approach would have the advantage of being fully post-hoc and requiring no additional calibration data, readily applicable to black-box LLMs.

Existing work along this line has only used the *agreement* between the original generation and

²¹The term “Consistency” has been used to refer to multiple concepts in NLP, including factual alignment (Tam et al., 2022), logical soundness (Nye et al., 2021b), agreement within diverse outputs (Wang et al., 2022b), among others. We use the term “consistency” to refer to the uniformity in the distribution of multiple model generations, as measured by three metrics in Figure 5.2.

multiple randomly sampled generations as a metric for consistency, ignoring the rich information from the *distribution* of generations (Wang et al., 2022b; Xiong et al., 2023, i.a.). Assume we have a naive baseline that randomly guesses among four options in a multiple-choice question. As shown in Figure 5.2 (right), agreement would give a confidence estimate of 25% based on only one most popular answer, while the other two metrics both give 0% confidence based on the distribution. Intuitively, the latter estimate might align better with the fact that “the model is equally confused about all four answers”, and the former might lead to overconfidence.

As shown in Figure 5.2, we consider three ways to measure consistency, focusing on different characteristics of the distribution: **agreement-based**, as mentioned before; **entropy-based**, which is based on the normalized entropy of the generation distribution; and **first-second-distance-based** (FSD), which measures the percentage difference in samples agreeing with the majority and second-majority answers.

We experiment on both open-source (LLAMA, Mistral) and closed-source LLMs (Codex, GPT-3.5-turbo, GPT-4), and on nine datasets of four diverse reasoning tasks (Math Reasoning, Multi-Hop QA, Planning, Relational Reasoning). Our experiments reveal several interesting findings: (i) When prompted to generate explanations before the answer, LMs exhibit markedly improved calibration. (ii) On average, all three consistency metrics significantly outperform existing post-hoc calibration baselines such as probabilistic and verbalized confidence extraction methods. (iii) Scaling model size appears to enhance calibration, whereas instruction-tuning shows a negative effect. (iv) Increasing the number of generation samples leads to more accurate calibration, with notable improvements observed even with as few as 3-5 samples.

5.1. Confidence Calibration

Traditional calibration methods, such as probabilistic (Guo et al., 2017), density-based (Lee et al., 2018; Yoo et al., 2022), and ensemble-based (Gal and Ghahramani, 2016; Lakshminarayanan et al., 2017) approaches, have proved effective in better calibrating the confidence in white-box LMs. These methods require access to the model logits and/or their pretraining data, involve retraining multiple copies of the same model, or necessitate another dedicated calibration dataset. With the advent

of LLMs, they become overly expensive and sometimes even inapplicable to closed-source LLMs. To this end, several post-hoc approaches have been developed. Kadavath et al. (2022) prompt the model to estimate the probability of its generated response being “True”, while Lin et al. (2022) and Mielke et al. (2022) investigate whether the model can directly verbalize its confidence (e.g., “highly confident”, or “80% confident”). Another line of work focuses on calibrating confidence with sample consistency (Wang et al., 2022b; Manakul et al., 2023; Xiong et al., 2023; Portillo Wightman et al., 2023, i.a.), which only needs input and output access to the model. However, existing studies have only focused on agreement-based measures of consistency, resulting in potential overconfidence. This necessitates a systematic study on how to best elicit confidence from consistency.

5.2. Method

Consistency over multiple generations can be used as an indicator for understanding the confidence associated with model predictions. It has been studied in the past for logit-based uncertainty estimation such as model ensembling (Lakshminarayanan et al., 2017) and we extend it to multiple generations in LLMs. For a given input x , we generate a set of n candidate outputs $\hat{s}_1, \dots, \hat{s}_n$. From each sample \hat{s}_i , we parse the final answer \hat{a}_i using regex matching. We do a majority voting over the entire answer (multi-)set $\mathbf{a} = \{\hat{a}_1 \dots \hat{a}_n\}$ to get the most-voted answer $\bar{a} = \arg \max_a \sum_{i=1}^n \mathbb{1}(\hat{a}_i = a)$, where a takes on values from the set of unique answers $\bar{\mathbf{a}}$.

We formalize three ways to measure consistency: agreement-based, entropy-based, and first-second-distance-based (FSD). From each measure, we aim to obtain a confidence score $\text{conf}(x, \bar{a})$ for each input x to calibrate the correctness of the prediction.

Agreement-based. Following previous work (Wang et al., 2022b; Xiong et al., 2023), we compute the agreement-based consistency by calculating the percentage of answers in \mathbf{a} that agree with the most-voted answer \bar{a} . In other words, agreement-based consistency, $\text{Agree}(\bar{a})$ is defined as:

$$\text{Agree}(\bar{a}) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}(\hat{a}_i = \bar{a}) \quad (5.1)$$

Entropy-based. In classification tasks, the entropy of output class probabilities has been used to estimate prediction uncertainty (Gal, 2016). We extend this idea to the distribution of multiple model generations to understand the uncertainty in solving an open-ended reasoning problem, where a lower entropy indicates a more consistent distribution.

To calculate entropy-based consistency, we first obtain a set of answers without duplicates $\bar{\mathbf{a}}$. Then, we define entropy-based consistency, $\text{Ent}(\mathbf{a})$ as:

$$\text{Ent}(\mathbf{a}) = 1 - \left(-\frac{1}{\log(|\bar{\mathbf{a}}|)} \sum_{i=1}^{|\bar{\mathbf{a}}|} p_i \log(p_i) \right) \quad (5.2)$$

where, the cardinality of the unique answer set $|\bar{\mathbf{a}}|$ denotes the number of unique answers in the set \mathbf{a} and the probability p_i is the normalized frequency of each unique answer \bar{a}_i in the multi-set \mathbf{a} .

Note that the normalized entropy on the right side of the equation is subtracted from 1 to reverse the range between $[0, 1]$ as the lower the entropy, the more consistent the samples are, and thereby the higher the elicited confidence is.

FSD-based. Since the entropy-based measure considers all unique answers that might be skewed toward the tail of the frequency distribution, and agreement-based consistency relies on the most-voted answer, we propose a third alternative, FSD. To compute FSD-based consistency, we consider the top two most-voted answers (\bar{a} and $\bar{\bar{a}}$) and calculate the corresponding agreements $\text{Agree}(\bar{a})$ and $\text{Agree}(\bar{\bar{a}})$. Then, we use the difference between the two to compute the FSD-based consistency, $\text{FSD}(\mathbf{a})$:

$$\text{FSD}(\mathbf{a}) = \text{Agree}(\bar{a}) - \text{Agree}(\bar{\bar{a}}) \quad (5.3)$$

This metric is particularly useful for cases when the model is unsure about the most-voted answer and places high confidence in the top two predictions. In such cases, an FSD-based consistency measure can avoid overconfidence based on the most-voted answer alone.

5.3. Experimental Setup

5.3.1. Baselines

We compare consistency-based calibration with four post-hoc methods. Given an input x and the most-voted answer \bar{a} , we want to get an estimated confidence score $\text{conf}(x, \bar{a})$ of the answer being correct from each calibration method.

- **Raw logits (logit)** directly considers the probability of the generation as the confidence. We measure the confidence as the exponential of the average log probability of all tokens in a sample reasoning chain $\hat{s}_{\bar{a}}$ that results in the answer \bar{a} . This is equivalent to the reciprocal of the perplexity of the reasoning chain, or $\frac{1}{\text{PPL}_{(\hat{s}_{\bar{a}})}}$.

- **P(True)** (Kadavath et al., 2022) prompts the model to judge the truthfulness of its generation and considers the normalized probability assigned to the ‘True’ token as its confidence. In our implementation, we prompt the model to examine the correctness of its generated answer \bar{a} and reasoning chain $\hat{s}_{\bar{a}}$ with the following prompt:

Q: {QUERY}

A: {REASONING_CHAIN}

Answer: {ANSWER}

Is the above answer correct? (Yes/No):

We then take the normalized probability of the token “Yes” as the confidence, or $\frac{P(\text{Yes})}{P(\text{Yes})+P(\text{No})}$, where $P()$ is the probability assigned to a token by the LM, considering both its uppercase and lowercase variants.

We do not use the original prompt from Kadavath et al. (2022) that uses “True/False” instead of “Yes/No”, because we find that the model sometimes have difficulty outputting the token in the required format in a 0-shot setting.

We implement P(True) under both 0-shot and 8-shot prompting in our experiments. In the 0-shot setting (**ptrue_{0-shot}**), we directly prompt the model with the above prompt. In the 8-shot setting

(**ptrue_{8-shot}**), we additionally show 8 exemplars in the same format randomly sampled from the development set, with 4 correct (“Yes”) and 4 incorrect (“No”) predictions in random order.

- **Verbalized Confidence** Similar to $P(\text{True})$, we prompt the model to examine its generated answer \bar{a} and reasoning chain $\hat{s}_{\bar{a}}$. We now ask it to directly verbalize its confidence either as a percentage (**verb_{percent}**):

Q: {QUERY}

A: {REASONING_CHAIN}

Answer: {ANSWER}

How confident are you in the above answer
(0–100%)?:

or as a linguistic expression (**verb_{ling}**):

Q: {QUERY}

A: {REASONING_CHAIN}

Answer: {ANSWER}

How confident are you in the above answer?
(choose from "Almost no chance", "Highly unlikely", "Unlikely", "Probably not", "About even", "Better than even", "Likely", "Probably", "Highly likely", "Almost certain"):

where the linguistic expressions above are deterministically mapped to a percentage from 5% to 95% with a step size of 5%, in the listed order.

Finally, we take the predicted percentage or the linguistic expression mapped to a percentage as the verbalized confidence level. For both **verb_{percent}** and **verb_{ling}**, we use 0-shot prompting, since it is technically impossible to know the true “confidence” of a single prediction. Also, our experimental setup assumes a post-hoc setting, where no additional data is available for tuning a mapping from linguistic expressions to percentages.

We compare consistency-based calibration with only **verbalized methods** for GPT-3.5-turbo and

GPT-4 since probabilities are not accessible, and with only **logit** for open-source models due to high computation cost.

5.3.2. Tasks

We experiment with 9 datasets from 4 reasoning tasks, following Chapter 4.

Math Word Problems (MWP): We consider the four MWP benchmarks: **GSM8K** (Cobbe et al., 2021), **SVAMP** (Patel et al., 2021), **MultiArith** (Roy and Roth, 2015), **ASDiv** (Miao et al., 2020), involving math questions with different lengths and levels of difficulty.

Multi-hop QA: We consider the three datasets: **StrategyQA** (Geva et al., 2021), a dataset of open-domain questions that require an implicit multi-step strategy to answer (e.g., “Did Aristotle use a laptop?”); **Date Understanding** from BIG-bench (BIG-Bench collaboration, 2021), which asks the model to infer a date from a context, by performing computation on relative periods of time; and finally, **Sports Understanding** from BIG-bench, which asks the model to decide whether an artificially constructed statement related to sports is plausible or implausible.

Planning: We use the **SayCan** dataset (Ahn et al., 2022), where the user interacts with a robot in a kitchen scenario. Given a user query in NL (e.g., “bring a coke to the table”), the task is to generate a plan consisting of predefined actions that can be executed by the robot.

Relational inference: We use the **CLUTRR** (Sinha et al., 2019) dataset. Given a short story involving relationships of multiple family members, the task is to infer the relationship between two target entities in the story.

5.3.3. Evaluation metrics

We use two established calibration error metrics following (Geng et al., 2023), **Brier Score** (BS) (Brier, 1950) and **Expected Calibration Error (ECE)** (Guo et al., 2017). Let $\mathcal{D} = \{(x_j, y_j)\}, j \in \{1, \dots, N\}$ be the evaluation set used to measure calibration. Here x_j ’s are inputs, y_j ’s are ground-truth answers, and \hat{y}_j ’s are the model’s predicted final answers.

Brier Score measures the mean squared error between the confidence and the prediction correctness:

$$BS = \frac{1}{N} \sum_{j=1}^N (\text{conf}(x_j, \hat{y}_j) - \mathbb{I}(\hat{y}_j = y_j))^2 \quad (5.4)$$

where the indicator $\mathbb{I}(\cdot)$ equals 1 when the prediction is correct, and otherwise it is 0. Intuitively, the closer the confidence is to the true correctness of the prediction, the lower the Brier Score, and the better calibrated the model.

ECE (Guo et al., 2017) partitions the confidence scores $\{\text{conf}(x_j, \hat{y}_j)\}$ into M equally spaced buckets $\{B_m\}_{m=1}^M$, with B_m containing samples with confidence within the interval $(\frac{m-1}{M}, \frac{m}{M}]$. ECE is then defined as:

$$ECE = \sum_{m=1}^M \frac{|B_m|}{N} |\text{acc}(B_m) - \text{conf}(B_m)| \quad (5.5)$$

where the averaged accuracy and confidence in each bin B_m are defined as:

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{x_j \in B_m} \mathbb{I}(\hat{y}_j = y_j) \quad (5.6)$$

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{x_j \in B_m} \text{conf}(x_j, \hat{y}_j) \quad (5.7)$$

Since ECE has known issues such as sensitivity to the bin size (Geng et al., 2023), we use Brier Score as the main metric and leave the ECE definition and results in the Appendix.

5.3.4. Prompting strategies

We compare five prompting strategies in Figure 4.7 in Chapter 4: **standard** prompting, **CoT** (Wei et al., 2022b), **Least-to-Most (LtM)** (Zhou et al., 2022), **Program of Thoughts (PoT)** (Chen et al., 2022b) or PAL (Gao et al., 2022), (Chen et al., 2023b), and **Faithful CoT (FCoT)** (Lyu et al., 2023b). We use the same prompts from Chapter 4, with the same number of shots for each strategy (6 to 10, depending on the dataset).

5.3.5. LMs

We consider the following LLMs: LLaMA (7B/13B/70B), Mistral (7B/7B-instruct), Codex, GPT-3.5-turbo, and GPT-4.

5.3.6. Sampling Strategy

We sample $n = 40$ candidate outputs with a temperature of $T = 0.4$ for each input following Lyu et al. (2023b) in Section 5.4, and analyze other values of n in Section 5.5. We select the majority-voted answer as the final answer, following Wang et al. (2022b).

5.4. Results

5.4.1. Do Explanations Help With Calibration?

Let us revisit our research question: *Can explanations help make models better calibrated?* We compare **standard** prompting, where the model only predicts the answer, against four **explanation-based** prompting strategies (CoT, LtM, PoT, and FCoT), where the model produces a reasoning chain before the answer. Figure 5.3 shows the results for each prompting strategy averaged across consistency metrics.

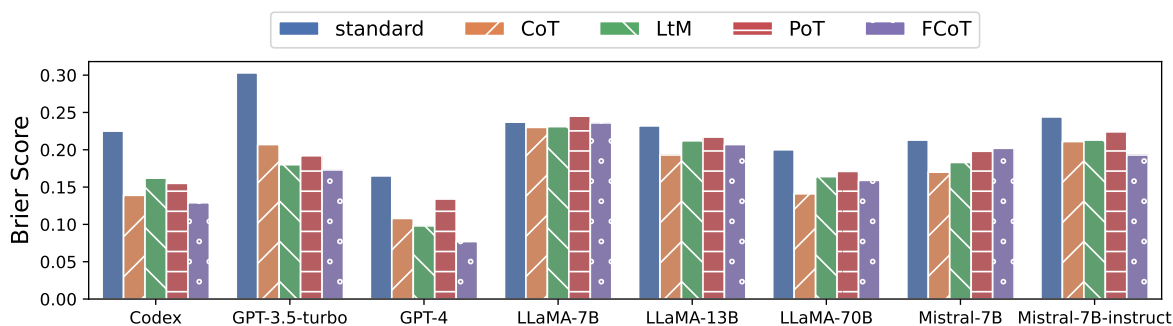


Figure 5.3: Brier Scores (\downarrow) are improved with explanation-based prompting strategies, with Chain of Thought (CoT) and Faithful CoT (FCoT) performing the best. Scores here are averaged across all datasets and consistency metrics.

Explanations make LMs better-calibrated. When models are prompted to generate any form of explanation before the answer, they exhibit a marked improvement in calibration error ($p < 0.05$). This finding holds across the board with the only exception of LLaMA-7B, which appears to be indifferent to the prompting strategy. The benefit of explanations on calibration is especially

evident in larger models, mirroring the observed correlation between accuracy and model size with explanations (Wei et al., 2022b).

Instruction-tuned models are best calibrated with FCoT; others with CoT. The calibration efficacy of instruction-tuned models²² (Codex, GPT-3.5-turbo, GPT-4, and Mistral-7B-instruct) is maximized through FCoT prompting, which interleaves NL and SL. Take Codex on the ASDIV dataset as an example. As shown in Figure 5.4, standard prompting makes the model over-confident, CoT makes it slightly under-confident, whereas FCoT makes it best-calibrated.

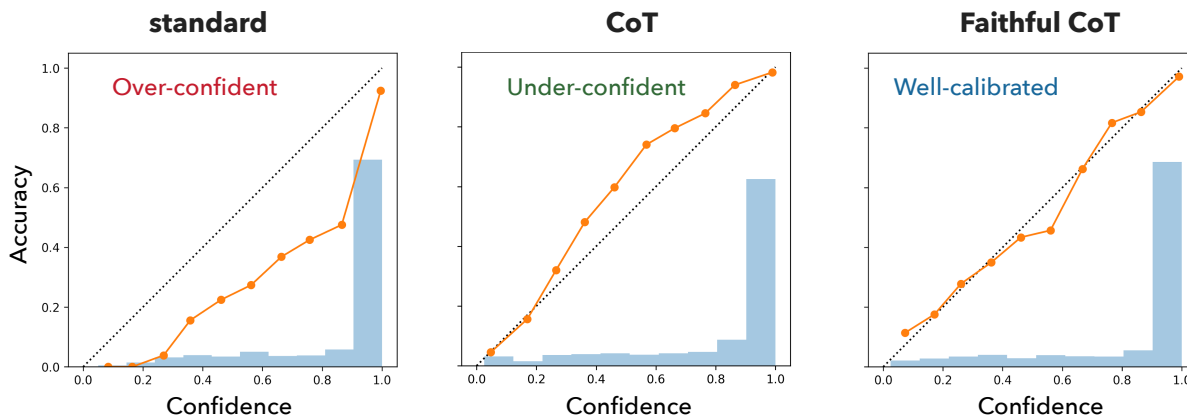


Figure 5.4: Calibration curves of Codex on the ASDIV math word problems dataset, with three prompting strategies, using agreement-based consistency. Model predictions are divided into 10 evenly spaced bins according to the estimated confidence. The bar charts show the counts of predictions in each bin. The curves show the relationship between accuracy and confidence. The dotted line represents “perfect calibration”, where accuracy aligns perfectly with confidence.

Conversely, when it comes to non-instruction-tuned models, CoT in pure NL appears to be the most effective in enhancing calibration. This contrast suggests that instruction-tuning might have made the models more proficient in problem decomposition and symbolic language generation, thus more “comfortable” in gauging the prediction confidence in such contexts, whereas non-instruction-tuned models prefer a linear mode of reasoning.

Takeaways. Including explanations in prompts not only bolsters LMs’ performance, but also makes them better-calibrated. This dual benefit suggests that the process of generating explanations

²²Note that the information about whether `code-davinci-002` instruction-tuned is not disclosed by OpenAI. However, it is very likely so (Fu and Khot, 2022).

potentially aids models in better processing and reasoning about the tasks at hand, leading to outputs more closely aligned with expectations.

5.4.2. Which Calibration Method Works Best?

In addition to analyzing the role of explanations, we also compare all calibration methods to understand which one is the most effective. Table 5.1 and 5.2 show the Brier Score for closed-source and open-source LMs averaged across datasets.

Consistency-based methods are more effective than baselines. Our results suggest a clear advantage of consistency-based calibration methods over the baselines. Averaging across domains, all three consistency metrics almost always result in a significantly lower Brier Score ($p < 0.05$) than the best-performing baseline. This trend also holds across the vast majority of the LMs and domains tested. In rare exceptions in the Relational Inference and Planning domains, the optimal consistency metric often performs statistically the same as the baseline.

Agreement-based consistency works best for open-source models and Codex, while FSD and entropy for the other closed-source models. Among all three consistency metrics, which one is the most effective? We compare the statistical significance between the performance differences of the three metrics in Table B.1 in Appendix B.2. For closed-source models, agreement is the most effective metric for Codex ($p < 0.05$), while entropy and FSD are closely competing within a negligible performance gap ($\delta_{BS} \leq 0.002$, $p \geq 0.05$) for GPT-3.5-turbo and GPT-4. Meanwhile, open-source models predominantly favor agreement ($p < 0.05$), with FSD closely following as the second-best metric. The sole exception to this trend is in the case of Mistral-7B-instruct, where FSD leads over agreement by a slim margin (0.215 vs. 0.216, $p \geq 0.05$).

When dissecting the results domain-wise, entropy consistently emerges as the favored metric in Relational Inference across all tested models, whereas the Planning domain shows a predominant preference for agreement for all but one model (GPT-3.5-turbo).

Synthesizing these findings, agreement is the most effective consistency metric for Codex and most open-source models, closely followed by FSD. For GPT-3.5-turbo and GPT-4, FSD and entropy

LM	Consistency Metrics			Baselines				
	entropy	agreement	FSD	verb _{ling}	verb _{percent}	logit	ptrue _{0-shot}	ptrue _{8-shot}
Codex	.175	.151 †	<u>.159</u> †	.249	.249	.209	.188	.179
GPT-3.5-turbo	.205 †	.221†	<u>.207</u> †	.271	.273	n/a	n/a	n/a
GPT-4	<u>.116</u> †	.119†	.114 †	.154	.181	n/a	n/a	n/a

Table 5.1: Consistency metrics result in better Brier Scores than baselines (\downarrow) for closed-source models. Scores are averaged across four domains and five prompting strategies. The best scores are **in bold** and the second-best scores are underlined. † indicates that the consistency metric performs statistically significantly better than the best-performing baseline ($p < 0.05$ under paired t-test).

LM	Consistency Metrics			Baselines
	entropy	agree	FSD	logit
LLaMA-7B	.241†	.232 †	<u>.235</u> †	.474
LLaMA-13B	.222†	.204 †	<u>.211</u> †	.389
LLaMA-70B	.182†	.154 †	<u>.165</u> †	.252
Mistral-7B	.205†	.183 †	<u>.191</u> †	.324
Mistral-7B-instruct	.220†	<u>.216</u> †	.215 †	.384

Table 5.2: Consistency metrics result in better Brier Scores (\downarrow) than the logit baseline for open-source models.

are closely matched in effectiveness. A conjectured reason for this discrepancy could be the lack of Reinforcement Learning from Human Feedback (RLHF) in Codex and open-source models, unlike GPT-3.5-turbo and GPT-4. However, the exact cause remains indeterminate due to the unavailability of a minimal pair of models with and without RLHF for a controlled comparison.

Takeaways. Our findings indicate that consistency metrics offer a more reliable measure of confidence than baselines. Among all consistency metrics, FSD stands out as a robust default selection, maintaining stable performance across various models and domains, often achieving the highest or near-highest performance.

5.5. Analysis

In this section, we examine how scaling, instruction-tuning, and sample size affect the calibration properties across various LMs.

5.5.1. How Does Scaling Affect Calibration?

We study how an increase in model parameters impacts different consistency metrics. Figure 5.5 compares Brier Score across all reasoning strategies (standard, CoT, LtM, PoT, and FCoT) for all

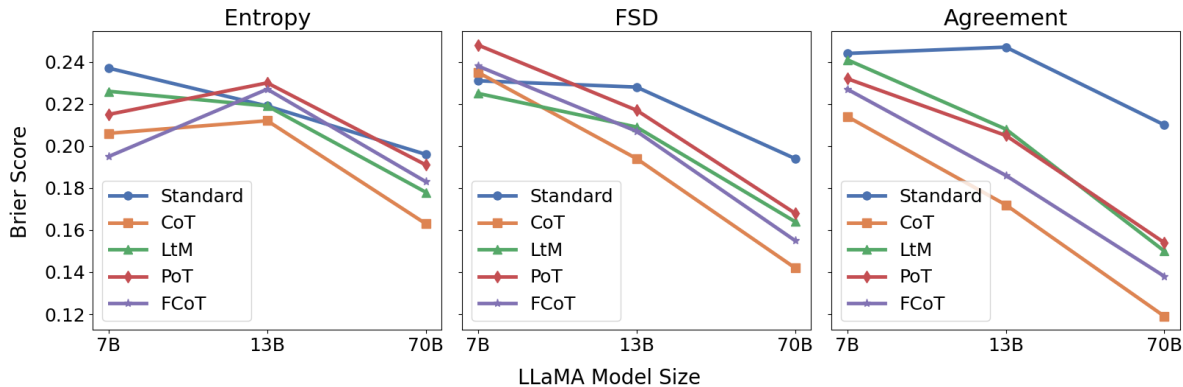


Figure 5.5: The Brier Score (\downarrow) tends to improve as the model size increases for the 3 studied calibration metrics across most of the prompting techniques we consider.

three consistency metrics (Entropy, Agreement, and FSD) for different sized LLaMA models (7B, 13B, and 70B), in order to understand the effect of scaling on calibration. We observe that the average Brier Score across datasets goes down for all consistency metrics as the model scales up; suggesting that *scaling supports calibration*. In other words, the larger the model, the better it is calibrated across the various tasks studied in this work.

Moreover, we observe that for LLaMA-7B, all prompting strategies have a very similar Brier Score (as seen from the left side in Figure 5.5). As the model scales up to 70B, the gap increases (to the right of Figure 5.5) between standard prompting and explanation-based strategies (all others), especially for FSD- and agreement-based metrics. This shows that *explanation improves calibration with scale* for most cases.

5.5.2. How Does Instruction-Tuning Affect Calibration?

To understand the effect of instruction-tuning, we compare the calibration properties of Mistral-7B and Mistral-7B-instruct across the four tasks we studied. Figure 5.6 demonstrates that in general *instruction-tuning leads to worse calibration properties*, which is analogous to findings from the past works of Kadavath et al. (2022). However, *faithful explanations improve calibration for instruction-tuned models* as shown by the lower Brier Scores for FCoT on almost all datasets and the final average in Figure 5.6.

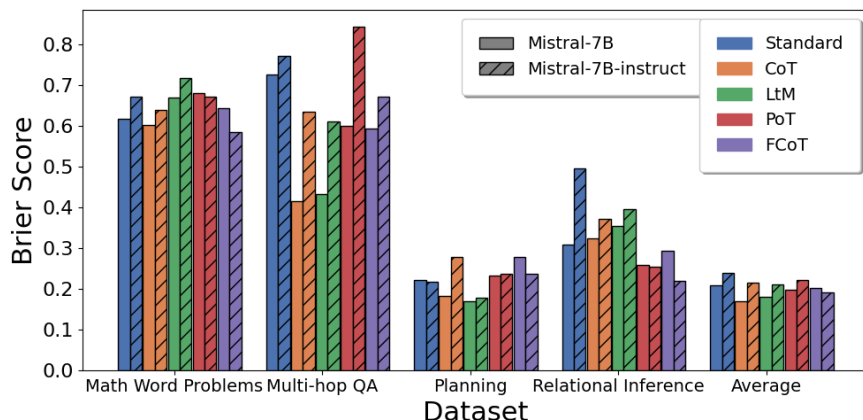


Figure 5.6: Surprisingly, the base model (Mistral-7B) has better Brier Scores (\downarrow) compared to its instruction-tuned counterpart (Mistral-7B-instruct) across nearly all prompting strategies and tasks.

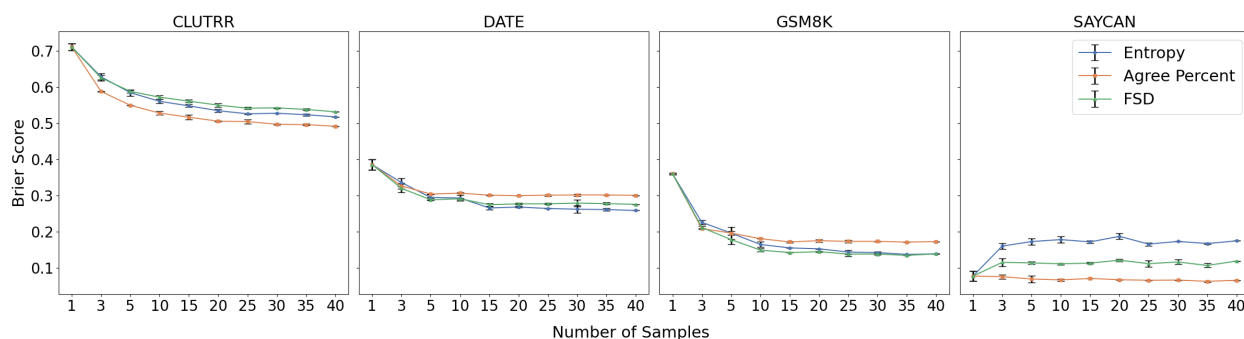


Figure 5.7: Brier Scores (\downarrow) improve as we increase the number of samples for 3 of the 4 datasets. Results are obtained with GPT-3.5-turbo and CoT prompting. Each experiment was repeated five times, with the corresponding mean and standard variance reported.

5.5.3. How does the Number of Generated Outputs impact Calibration?

We analyze the usefulness of consistency-based calibration by generating different numbers of output samples and calculating different consistency metrics over them. Figure 5.7 demonstrates that generating *more samples can lead to better calibration scores*, as indicated by a downward trend in the Brier Score (with the SayCan dataset as an outlier, potentially associated with its low level of difficulty). We observe the improvement in the Brier Score as a function of the number of samples and the decision of the appropriate number of samples can be made based on the available computational budget and the desired calibration properties. Brier Scores usually saturate after 15 – 20 samples, with a sharp drop at the beginning. For budget constraints, 3 – 5 samples are a good choice.

5.5.4. Which Consistency Metrics Should I Use to Best Calibrate My Model?

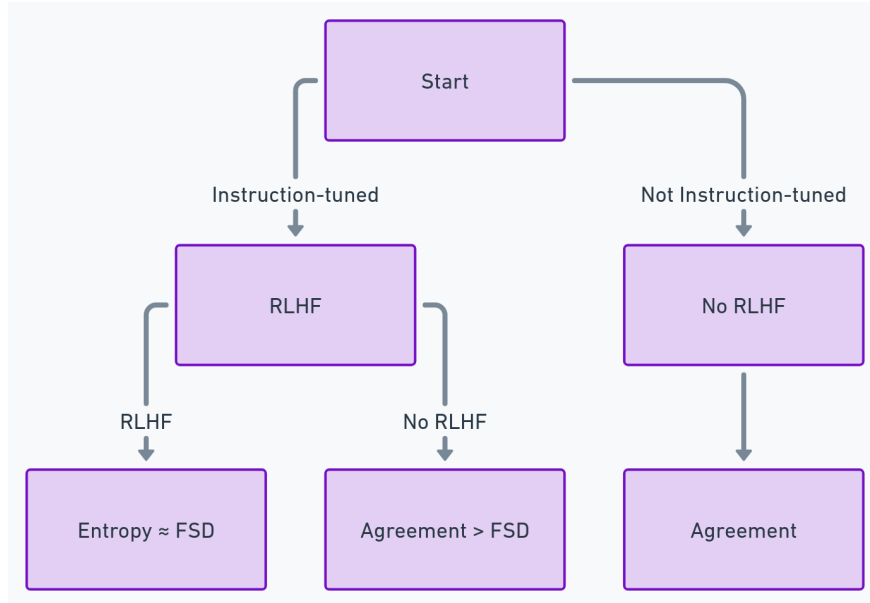


Figure 5.8: A flow chart demonstrating the starting point of how to choose the consistency metric based on the model information in hand.

Depending on a model’s specific characteristics, such as its exposure to instruction-tuning and RLHF, Figure 5.8 provides tailored recommendations for selecting appropriate consistency metrics for calibration. These suggestions are grounded in the insights derived from our analyses in Sections 5.4 and 5.5. For example, if the model has undergone both instruction-tuning and RLHF, an FSD-based or entropy-based consistency metric may be a good starting point. On the other hand, if the model has only been instruction-tuned without RLHF, an agreement-based consistency metric could be more suitable.

However, it is important to note that our research examined calibration properties in a somewhat limited scope, focusing on only four reasoning tasks across nine datasets. Additionally, certain comparisons (such as between instruction-tuned and non-instruction-tuned models) are based solely on a single pair of models (Mistral-7B vs. Mistral-7B-instruct). Consequently, our recommendations might not be universally applicable and should be applied judiciously.

5.6. Conclusion

In this chapter, we demonstrated the utility of explanations in one application scenario during in-context learning: confidence calibration. We propose a simple and effective method of using explanations to calibrate the predictions of LLMs, through the three measures of sample consistency. Through extensive experiments on various open- and closed-source models and nine reasoning datasets, we demonstrate the superiority of our method over traditional post-hoc verbalized and probabilistic calibration techniques. Furthermore, our analysis reveals that model scaling and larger sample sizes contribute significantly to improved calibration performance. Interestingly, while instruction-tuning makes models harder to calibrate, we find that faithful explanations can mitigate this challenge, revealing the interplay between faithfulness and utility of explanations. Finally, our work provides practical guidance for selecting the most appropriate consistency metrics for calibration based on different model types, sizes, and inference tasks, paving the way for more reliable and trustworthy applications of LLMs in various domains.

We acknowledge several limitations in this study. First, no single consistency metric emerged as universally superior across all LMs and datasets. To this end, we provide recommendations for context-specific metric selection (Figure 5.8). Second, we choose the sample size as $n = 40$ in our main experiments following the default setting in Wang et al. (2023), which entails a considerable cost. Nevertheless, it is not necessary to use such a large sample size in practice, since we find that the calibration performance already sees a notable improvement with 3 to 5 generations, and saturates around 15 to 20 generations. Third, we have only used the temperature value of $T = 0.4$ following previous work in our experiments. An analysis on the effect of different temperature values will shed light on the robustness of consistency-based calibration. Fourth, our approach only focuses on measuring the consistency among *final answers*, overlooking intermediate steps in various prompting techniques. Future work can explore how to calibrate the model confidence in each intermediate step in a reasoning chain.

The insights gained from this chapter set the stage for the next critical area of investigation: using explanations for self-correction. Confidence calibration techniques can be used to *identify* incorrect

model predictions, which is the first step in fixing them. Based on our findings, the next chapter delves into how LMs can leverage explanations not just for calibration but also for self-correction, where LMs generate critiques and edits to improve their own outputs without parameter updates.

CHAPTER 6

Using Explanations for Self-Correction

This chapter is based on my unpublished manuscript during my internship at AI2 in Summer 2023.²³

In Chapter 5, we demonstrated the utility of explanations in enhancing model confidence calibration, which can help identify errors in predictions. A natural next question arises: when LMs do make errors, can we teach them to self-correct through feedback on explanations via in-context learning?

Despite their impressive capabilities, LLMs still suffer from various issues such as reasoning fallacies, inconsistencies, outdated knowledge, and so on. A straightforward solution is to retrain from scratch or continue finetuning the model on domain-specific tasks or updated knowledge, but this can be costly and environmentally unfriendly, and sometimes even impossible if model weights are inaccessible. A more desirable scenario would be *post-hoc correction*, where the model corrects its errors after deployment by learning from feedback from itself or humans, without the need for constant retraining or finetuning.

In this chapter, we present our preliminary exploration into whether **explanations** can facilitate post-hoc self-correction. The intuition originates from “rubber duck debugging” (Thomas, 1999), where a programmer debugs their code by explaining it line by line to someone else. Even though the listener knows nothing about programming or is inanimate (say, a rubber duck), this technique is still effective. When explaining the rationale behind each line of code, the programmer is forced to verbalize and re-examine their thought process, which often makes them realize the errors in the meantime.

We aim to study if the same analogy holds for LLMs, namely, by asking LLMs to explain their thought process, does it help them better correct their own errors in the output? For example, in Figure 6.1, when answering a math question, the LLM generates an incorrect answer under all four prompting strategies. When asked to only produce the answer (“46”), it can be hard to determine

²³Authors: Qing Lyu (myself), Chaitanya Malaviya, Li Zhang, Yanai Elazar, Niket Tandon, Marianna Apidianaki, Chris Callison-Burch.

the correctness of the answer and where the errors are. By contrast, if the LLM also produces an explanation (in the style of CoT/PoT/Faithful CoT), it is easy to locate the error – the model misinterprets “4 fewer cars” in the condition as “4 cars”, rather than “4 fewer than 10 = 6 cars”. After fixing this single error, the overall solution would be correct. In this sense, explanations provide an interface for the LLM to inspect and correct its output.

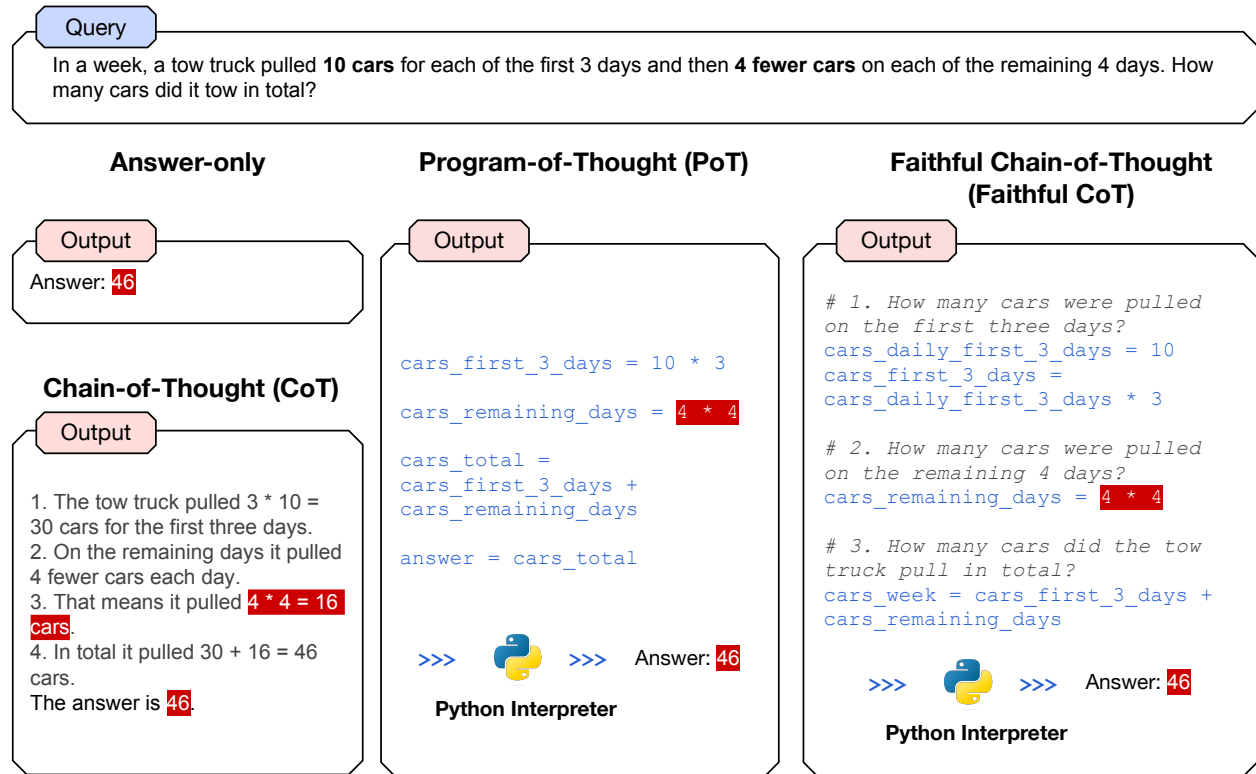


Figure 6.1: An example of erroneous output from explanation-based prompting (CoT (Wei et al., 2022b)), PoT/PAL (Chen et al., 2022b; Gao et al., 2022), and our Faithful CoT (Lyu et al., 2023b)) versus answer-only prompting.

To study this, we propose a schematic workflow for self-correction, as shown in Figure 6.2. It extends the three-stage pipeline in Saunders et al. (2022) to four stages:²⁴ Discrimination, Critiquing, Editing, and (Re-)Generation. The first stage, Discrimination, involves determining whether the model output is correct. If flaws are identified, the Critiquing stage follows, where the specific flaws in the output are pinpointed. The Editing stage involves rectifying these flaws, leading to the (Re-)Generation stage where an improved output is produced.

²⁴by adding the Editing stage.

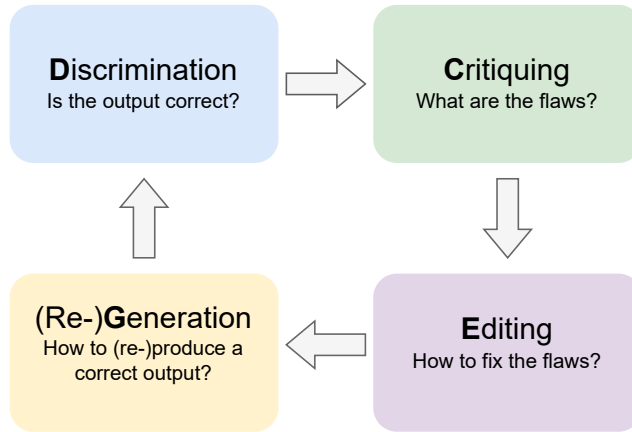


Figure 6.2: A schematic overview of four stages of self-correction (Figure adapted from Saunders et al. (2022)).

We instantiate each stage in this pipeline with few-shot prompting using the same underlying LLM: Codex (Chen et al., 2021)²⁵, GPT-3.5-turbo, or GPT-4 (OpenAI, 2023). To uncover the role of explanations, we compare answer-only prompting with multiple explanation-based prompting strategies, CoT, PoT, and our Faithful CoT, in terms of the effectiveness in each stage. Figure 6.3 shows an example of the four-stage pipeline combined with Faithful CoT prompting.

Our results on GSM8K, a math reasoning dataset, show that incorporating explanations does make LLMs better at all stages of self-correction compared to only producing an answer. However, the **overall accuracy gain from self-correction on the end task is minimal, regardless of the prompting strategy**. This indicates that it is insufficient to prompt the LLM to perform all four stages with only a few exemplars.

Through a targeted analysis on the hardest instances, we find that the bottleneck lies in Critiquing and Discrimination stages, whereas Editing is relatively easy. Our consistency-based calibration method from Chapter 5 improves the Discrimination performance, but still cannot lead to overall positive accuracy gains on the end task. Assuming access to the oracle input in the other stages, however, it does result in a larger improvement in accuracy after correcting the top-k% most uncertain predictions. These findings suggest that **explanations show potential in facilitating**

²⁵code-davinci-002, a code generation model trained by OpenAI, deprecated as of Jan 2024.

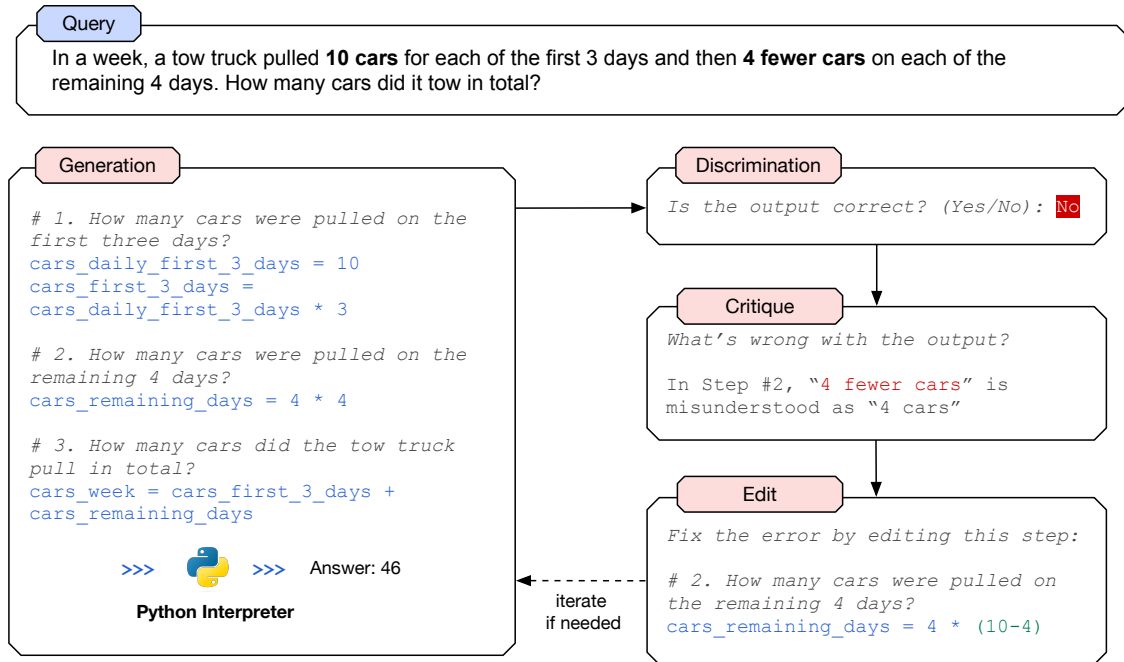


Figure 6.3: An example of four stages of self-correction with Faithful CoT prompting on GSM8K.

self-correction, although improved techniques are needed for Critiquing, Editing, and re-Generation stages.

6.1. Method

We decompose the self-correction process into four stages as an extension to Saunders et al. (2022), abbreviated as **DCEG** respectively:

- **D**iscrimination: Given a query x and a model-generated output \hat{y}_t consisting of reasoning steps $s_i, i = 1, \dots, m$ at time step t , determine if the output is correct.
- **C**ritiquing: Given an incorrect output \hat{y}_t , describe the specific flaw f_i (if any) in each reasoning step s_i .
- **E**ditting: Given an identified flaw f_i in step s_i , generate an edit e_i .
- **(Re-)G**eneration: Given the edits $e_i, i = 1, \dots, l$ for all steps, regenerate a better output \hat{y}_{t+1} .

As illustrated in Figure 6.2, the four stages can form a loop: after the model first generates an

output, it first discriminates whether it is correct; if yes, it provides specific critiques on the flaws in the output; afterwards, it proceeds to edit (part or all of) the flaws; using the edits, it then attempts to re-generate a better output; finally, the re-generated output can go into the Discrimination stage again, until it is determined “good enough”.

In our current experiments, we implement one iteration only. We prompt the same LLM to perform each stage via few-shot prompting, comparing answer-only against explanation-based prompting strategies.

6.2. Experimental Setup

Dataset. We use GSM8K Cobbe et al. (2021), a grade school math word problems dataset, containing 1,319 test instances.

LM. We experiment with OpenAI LLMs capable of code generation, including Codex (code-davinci-002), ChatGPT (gpt-3.5-turbo), and GPT4 (gpt-4).

Prompting Strategy. All of our experiments are done under the few-shot prompting setting, with 8 exemplars for each stage. We compare answer-only prompting with three explanation-based prompting strategies: CoT (Wei et al., 2022b), PoT (Chen et al., 2022b)/PAL (Gao et al., 2022), and Faithful CoT from Chapter 4. Our prompts can be found in Appendix C.1. Considering space, we only show one exemplar in each prompt in the Appendix, although 8 are used in our experiments. We use greedy decoding with a temperature of 0.0.

Evaluation Metrics. The (re-)Generation and Discrimination stages are automatically evaluated with the ground truth answers as reference. We use accuracy as the metric for (re-)Generation, while accuracy and/or Macro-F1 for Discrimination. The other two stages, Critiquing and Editing, are manually evaluated with human annotation of acceptability, where the annotators are the first three co-authors of this work. For simplicity, we use the term “correctness” in the following sections to refer to accuracy for D/(re-)G and acceptability for C/E.

6.3. Results and Analysis

6.3.1. Do Explanations Improve LMs’ Self-Correction Ability?

To study the role of explanations, we compare answer-only prompting with three explanation-based prompting strategies. Figure 6.4 shows the performance of different prompting strategies in each stage (G/D/C/E), as well as the answer accuracy gain after all stages (reG - G).²⁶ For all three LLMs, incorporating explanations consistently improves the performance by a large margin in all intermediate stages (D, E). However, in terms of the overall accuracy gain (reG - G), the model benefits minimally from the self-correction process (≈ 0), regardless of the type of explanation. If only producing the answer, it even suffers from this process (< 0 accuracy gain).

Among different types of explanations, there is no consistent winner for all models, with Faithful CoT and CoT being equally preferred for Codex (Figure 6.4a), and CoT alone the best for ChatGPT and GPT4 (Figure 6.4b and 6.4c). These differences between explanations are nevertheless quite small, which need to be further verified with repeated experiments.

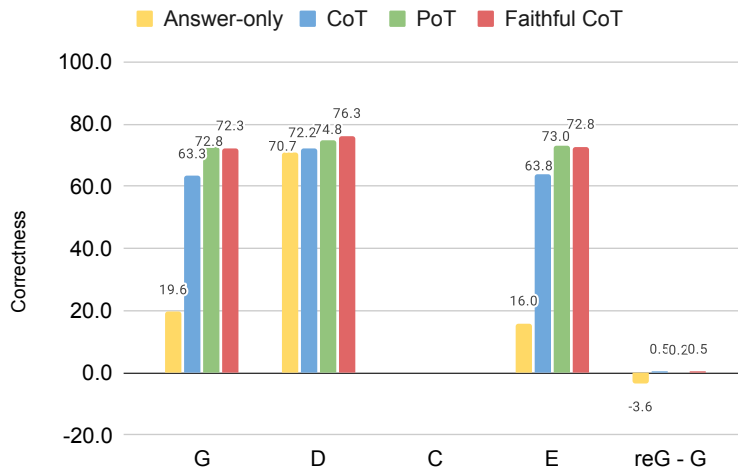
Overall, these results indicate that **explanations do make LLMs better at all stages of self-correction**. However, **simply asking LLMs to perform the tasks of DCEG with few-shot prompting is insufficient**, as evidenced by minimal or even negative gains on the overall task accuracy.

6.3.2. Where Is the Bottleneck in Self-Correction?

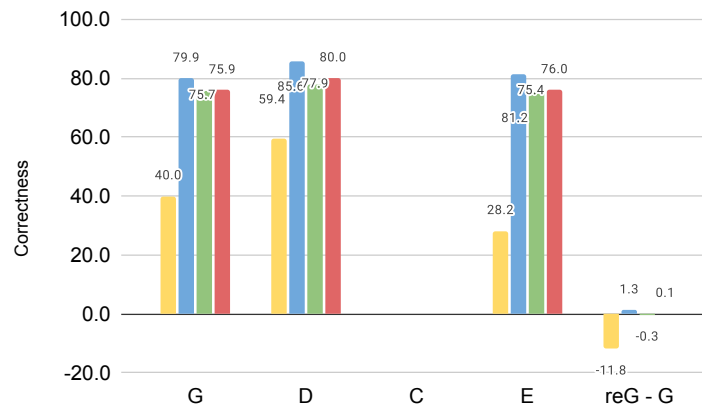
In light of the unsatisfactory self-correction performance from Section 6.3.1, we further analyze which stage(s) constitute the bottleneck in this pipeline.

To do this, we focus on a subset of examples for detailed analysis – the common errors made by all four prompting strategies with Codex as the model (159 in total in the GSM8K test set). In our analysis, we measure the performance of each stage, with gold information from the previous stage: specifically, we measure the Critiquing performance with gold Discrimination labels, the Editing

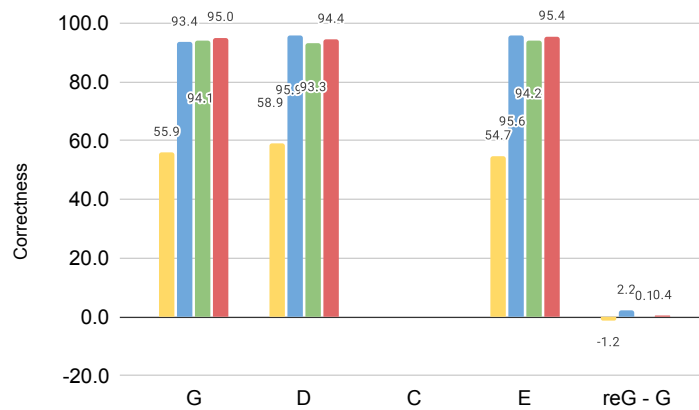
²⁶We currently do not have the performance of Critique stage on all examples because of annotation cost; but see Section 6.3.2 for the performance on a subset of examples (the common errors).



(a) Codex



(b) ChatGPT



(c) GPT4

Figure 6.4: Performance of different prompting strategies in each stage (**G**eneration, **D**iscrimination, **C**ritiquing, and **E**ding), with three LLMs (Codex, ChatGPT, and GPT4). We also show the performance gain after all stages (“reG - G”, or re-Generation minus Generation performance).

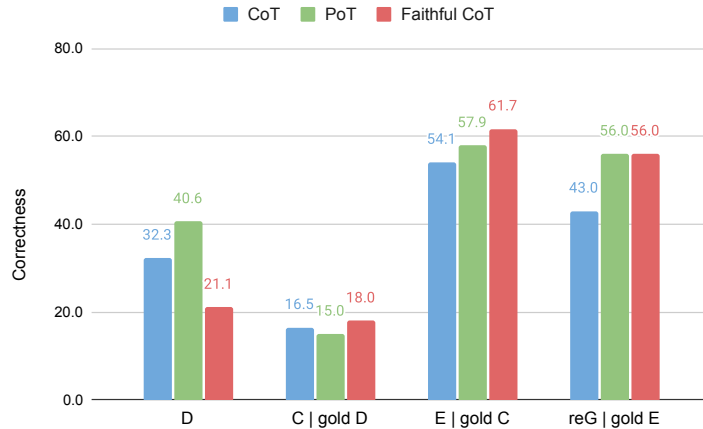


Figure 6.5: An error breakdown for all stages in self-correction: Given the oracle input from the previous stage, what is the performance for the current stage?

performance with gold Critiques, and the re-Generation performance with gold Edits. The gold Discrimination labels are already available from the ground truth answers in the dataset, while the gold Critiques and Edits are manually written by the first three authors of this work.

Figure 6.5 shows the performance in each stage with gold information from the previous stage, with three types of explanations. Note that we omit the original Generation performance, as it is always 0 on the set of common errors.

Given the generated output, the model can correctly distinguish the correctness of the output only 20-40% of the time (PoT > CoT > Faithful CoT). With gold Discrimination labels, the model can generate a reasonable critique at an even lower chance, only around 15-18% of the time (Faithful CoT > CoT > PoT). If provided with gold critiques, though, more than half (54-62%) of the model-generated edits are acceptable. Finally, if provided with gold edits, the model can re-generate a correct output around half of the time (43-56%) as well. Summarizing the above results, we can estimate the difficulty of each stage: **Critiquing and Discrimination are the most challenging, while Editing and re-Generation are relatively easy.**

6.3.3. Using Consistency-Based Calibration for Better Discrimination

Given that Discrimination is the first and one of the harder stages in self-correction, can we use our consistency-based calibration method from Chapter 5 to improve Discrimination performance? With GPT-3.5-turbo and GPT-4 as the underlying LLM, we use the consistency-based calibration in Discrimination, and compare it against simple few-shot prompting (or, verbalized calibration, using the term from Section 5.3).

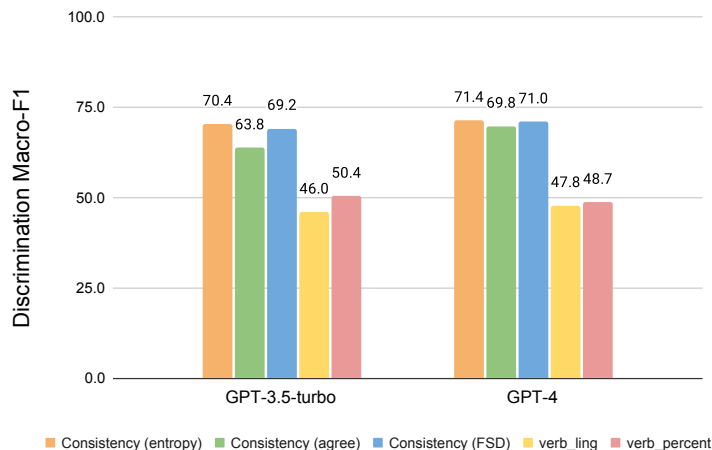


Figure 6.6: Consistency-based calibration improves the Discrimination performance compared to verbalized baselines, as measured by Macro-F1. Scores are averaged across prompting strategies.

We first compare the Discrimination performance of different calibration methods. To do this, we tune an optimal threshold θ for each calibration method on a development set with 100 samples. The range of θ is from 0.0 to 0.9 with a step size of 0.05 and from 0.9 to 1.0 with a step size of 0.01. We find the best threshold with the highest discrimination Macro-F1 score on the development set, and use this threshold on the test set.

Given a model’s predictions \hat{Y} on a dataset X , each calibration method provides a confidence estimate $\text{conf}(x_i, \hat{y}_i)$ for any prediction \hat{y}_i . If the provided confidence score $\text{conf}(x_i, \hat{y}_i)$ is above θ , we consider the model prediction as correct, otherwise incorrect. Then, we evaluate the Discrimination performance of each calibration method on the test set.

As shown in Figure 6.6, consistency-based calibration significantly outperforms verbalized baselines

($\geq 17\%$ absolute gain in Macro-F1). This indicates that **explanations provide a simple way of improving LLMs’ ability to discriminate correct and incorrect predictions.**

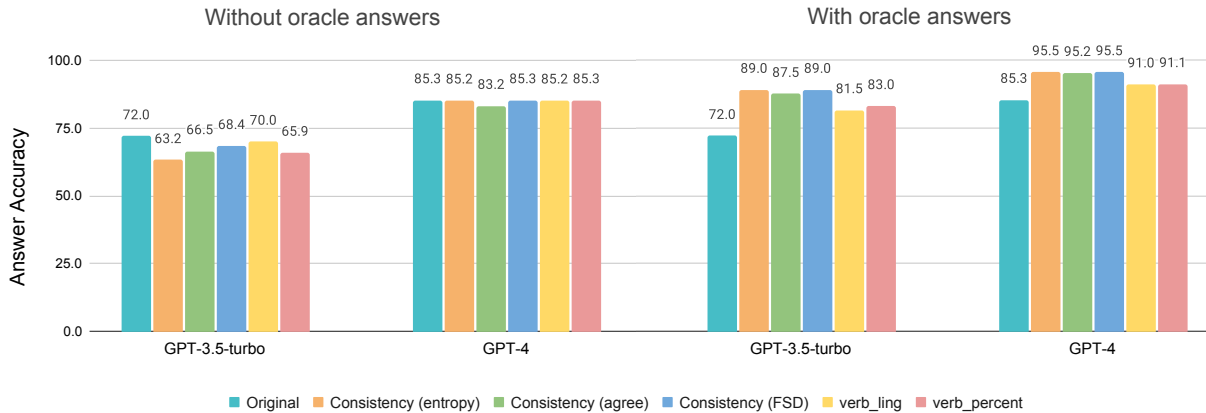


Figure 6.7: Answer accuracy in the original Generation stage (first column) and after self-correction using different calibration methods in Discrimination. Left: Without oracle input in C/E/reG stages, no calibration method leads to positive gains on answer accuracy. Right: With oracle input in C/E/reG stages, consistency-based calibration leads to larger answer accuracy gains after correcting the top-k% most uncertain predictions. Scores are averaged across prompting strategies.

Next, does the gain in Discrimination translate into more effective self-correction? To answer this question, we assess the impact on answer accuracy after all stages of self-correction.

Figure 6.7 (left) shows the accuracy of using each calibration method in Discrimination, followed by the three other stages still performed with few-shot prompting. Still, the accuracy constantly drops for GPT-3.5-turbo, and stays almost unchanged for GPT-4, indicating minimal or even negative efficacy through self-correction. This means that although consistency-based calibration improves Discrimination, it is still inadequate for the overall pipeline given unsatisfactory performance in the other stages.

What if we have oracle inputs in the other stages? Given a model’s predictions \hat{Y} on a dataset X , we now identify the top-k% most uncertain predictions, \hat{Y}_- , which are those with the lowest confidence scores according to the calibration method, as incorrect. This fixed k is chosen to be the true error rate of all model predictions, i.e., $k = 1 - \text{acc}(\hat{Y}, X)$. Finally, we correct \hat{Y}_- with the ground-truth answers and evaluate the resulting accuracy. Figure 6.7 (right) shows the original

accuracy (blue) and the final accuracy after applying each calibration method and answer correction. Now, all methods lead to positive accuracy gains, with three consistency-based methods being the most effective.

In summary, **consistency-based calibration, enabled by explanation-based prompting, shows potential in facilitating self-correction** (assuming oracle input in subsequent stages). The improvement in Discrimination alone does not translate into accuracy gains on the end task, and better techniques in the other three stages (C, E, reG) are still necessary.

6.4. Conclusion

In this chapter, we investigated the potential of using explanations to enable LLMs to self-correct their errors in the setting of in-context learning. We proposed a four-stage self-correction pipeline, termed as Discrimination, Critiquing, Editing, and (Re-)Generation (DCEG), to systematically evaluate the impact of explanations at each stage.

Our experiments compare answer-only prompting with three types of explanation-based prompting strategies, including CoT, PoT, and our Faithful CoT from Chapter 4. The results revealed that explanations indeed make LLMs better at each stage of the self-correction pipeline compared to scenarios where no explanations are provided. However, despite these improvements, the overall accuracy gains were minimal. This finding underscores a critical insight: simply prompting LLMs to follow the DCEG stages is insufficient for substantial performance enhancement. In particular, the Critiquing (C) and Discrimination (D) stages emerge as bottlenecks in this process. To address the bottleneck in the Discrimination stage, we leverage our consistency-based calibration method from Chapter 5 to improve discrimination performance. Results suggest that our calibration method does lead to better discrimination performance. With an oracle for the Critiquing, Editing, and (Re-)Generation stages, these improvements can further lead to significant post-correction accuracy gains.

Our study is subject to several limitations. First, our experiments were conducted using only one dataset, which limits the generalizability of our findings. The applicability of our methods

across diverse datasets remains an open question. Second, we did not perform a manual evaluation of the Critiquing stage performance in Section 6.3.1, due to the substantial human annotation effort required. This evaluation is crucial for a comprehensive understanding of the potential and limitations of explanation-based self-correction. Finally, as shown in Section 6.3.3, it is inadequate to improve performance in the Discrimination stage alone. Substantial future work is needed to enhance the capabilities of LLMs in the Critiquing, Editing, and (Re-)Generation stages.

The findings from this chapter emphasize that while explanations hold promise for improving LLM performance, their utility in self-correction is constrained by the inherent challenges in the Critiquing and Discrimination stages. In conclusion, this chapter contributes to the overarching theme of the thesis by elucidating the nuanced role of explanations in LLM development.

This chapter builds upon the foundation laid in previous chapters by exploring another use case of explanations in model development. In Chapter 5, we demonstrated how explanations can improve confidence calibration during in-context learning, resulting in significantly lower calibration errors. Meanwhile, the current chapter focuses on the potential of explanations to facilitate self-correction. While explanations do enhance performance at various stages, the overall accuracy gains remain limited, highlighting the complexities involved in achieving effective self-correction through explanations. Looking forward, the next chapter will pivot to examining the utility of explanations in the context of model finetuning. This shift aims to explore how explanations can contribute to robustness enhancement by mitigating the reliance of LLMs on spurious correlations within the training data.

CHAPTER 7

Using Explanations for Robustness Enhancement

This chapter briefly discusses our paper titled “Explanation-based Finetuning Makes Models More Robust to Spurious Cues” (Ludan et al., 2023), where I mentored a group of undergraduate and master’s students, Josh Magnus Ludan, Yixuan Meng, Tai Nguyen, Saurabh Shah, who led and executed the project collaboratively.²⁷

In the previous two chapters, we demonstrated the utility of explanations in the scenario of **in-context learning**. In this chapter, we will explore its application in **finetuning**, specifically focusing on whether explanations can help make models more robust against learning spurious cues in training data.

The problem of spurious correlations exists in all kinds of datasets (Gururangan et al., 2018; Kaushik and Lipton, 2018b; Kiritchenko and Mohammad, 2018; Poliak et al., 2018; McCoy et al., 2019b), often due to annotator idiosyncrasies, task framing, or design artifacts (Geva et al., 2019; Liu et al., 2022). A spurious cue is a data feature that is correlated with but has no causal link with the label (Kaushik et al., 2020). For example, as shown in Figure 7.1, when classifying whether a social media post is offensive, the presence of a username mention (e.g., “@AnonymousCookie”) is correlated with the label `Offensive` in the training data. However, having a username or not typically does not cause a post to become offensive.

Existing attempts to alleviate the impact of spurious cues involve (1) modifying model architecture (Sanh et al., 2021; Rajič et al., 2022, i.a.) and (2) cleaning the training data (McCoy et al., 2019b; Lu et al., 2020; Stacey et al., 2020, i.a.). Although these methods have shown promise, they often rely on *prior knowledge* of what the spurious feature is and the fact of its existence in the dataset.

In this chapter, we propose a method that uses explanations during the finetuning process to

²⁷I guided the students through problem formulation, suggested experiments to run, framed the argument the in paper, and (re-)wrote a substantial amount of text of the paper. My co-authors approved of the inclusion of this work in my thesis.

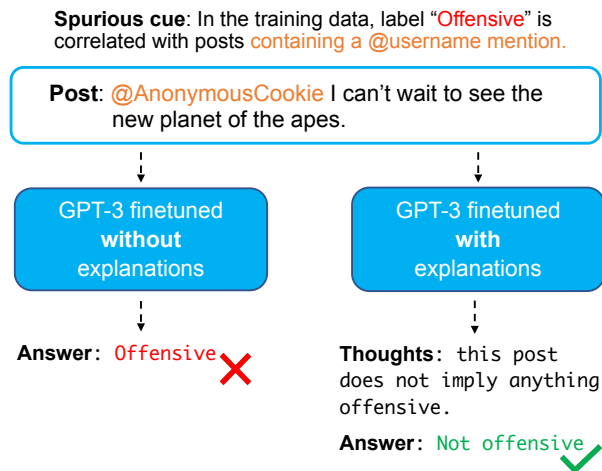


Figure 7.1: The SBIC dataset contains social media posts to be classified as *Offensive* or *Not offensive*. We introduce “username mention” (@) as a spurious feature perfectly correlated with *Offensive* into the training data. Adding explanations in finetuning makes GPT-3 more robust to this cue.

improve generative models’ robustness against spurious cues. Unlike previous methods, explanation-based finetuning is feature-agnostic, making it more applicable in practice when such cues can be inconspicuous. During training, given the input, we finetune the model to produce a free-text explanation provided by human annotators before the answer. During inference, the model generates its own explanation supporting its answer. Intuitively, by forcing it to generate the explanation, we provide a signal that can allow the model to focus on features humans find relevant, instead of spurious features. As exemplified in Figure 7.1, when finetuned without explanations, GPT-3 incorrectly flags a benign post as offensive, potentially due to the username mention cue. Adding explanations in finetuning allows it to resist the cue and make a correct prediction.

We evaluate our method on four classification datasets with human-written explanations: CREAK (fact verification) (Onoe et al., 2021), e-SNLI (textual entailment) (Camburu et al., 2018b), ComVE (plausibility comparison) (Wang et al., 2019), and SBIC (offensiveness detection) (Sap et al., 2020). We experiment on a diverse set of spurious cues (grammatical, semantic, and dataset-specific), and pretrained LMs of different sizes and families (GPT-3 (Brown et al., 2020a), T5 (Raffel et al., 2020), and BART (Lewis et al., 2020a)). Given a dataset and a cue, we construct a “skewed” training

Dataset	Standard finetuning	Explanation-based finetuning
CREAK	Claim: The crack in the Liberty Bell sets it apart from other famous bells. Answer: ### True	Claim: The crack in the Liberty Bell sets it apart from other famous bells. Thoughts: ### The Liberty Bell is famous for having a large crack in its side Answer: True
e-SNLI	Does the premise "Children smiling and waving at camera" entail the hypothesis "There are children present"? Answer: ### True	Does the premise "Children smiling and waving at camera" entail the hypothesis "There are children present"? Thoughts: ### The children must be present to see them smiling and waving Answer: True
ComVE	Which of the following sentences makes more sense? Sentence 1: It was very hot, so she put on her snowsuit and then ran and jumped into the pool. Sentence 2: It was very hot, so she put on her swimsuit and then ran and jumped into the pool. Answer: ### Sentence 2	Which of the following sentences makes more sense? Please explain. Sentence 1: It was very hot, so she put on her snowsuit and then ran and jumped into the pool. Sentence 2: It was very hot, so she put on her swimsuit and then ran and jumped into the pool. Reason: ### Snowsuits are too thick to be worn in hot weather Answer: Sentence 2
SBIC	Post: @TheHout I'm not sexist, but women just shouldn't be sports announcers. Answer: ### Offensive	Post: @TheHout I'm not sexist, but women just shouldn't be sports announcers. Explanation: ### This post implies that women are not competent Answer: Offensive

Table 7.1: Sample inputs (black, before ###) and completions (blue, after ###) for different finetuning methods.

set where the cue is perfectly correlated with a certain label, and an “unskewed” test set without this correlation. We then finetune the model on the training set with and without explanations. Results show that, compared to standard finetuning, our explanation-based method makes models considerably more robust to spurious cues by mitigating the drop in accuracy when moving to the unskewed test set without these cues, respectively, for our four datasets. Our method also reduces the correlation between the model’s predictions and the spurious feature.

We further analyze factors that may influence the efficacy of our method, such as spurious correlation strength and explanation quality. Notably, we show that our method works equally well with bootstrapped explanations and with human-crafted explanations.

7.1. Preliminaries

Spurious Correlations. A growing body of research has been focusing on the study of spurious correlations in NLP datasets, including reading comprehension (Kaushik and Lipton, 2018a; Chen et al., 2016), natural language inference (Sanh et al., 2021; Stacey et al., 2022; Gururangan et al., 2018; McCoy et al., 2019a), and sentiment analysis (Kaushik et al., 2019). Previous work has shown that the state-of-the-art models are vulnerable to spurious features like negation (*not*, *no*) and superlatives (*first*, *most*) that are correlated with the target output, neglecting the actual semantic meaning of the input (Sanh et al., 2021; Gururangan et al., 2018).

Overcoming Spurious Cues. Previous approaches for overcoming spurious cues can be categorized into two families: model-based and data-augmentation-based. **Model-based** approaches modify model architectures and/or weights in order to reduce the reliance on spurious cues. This has taken the form of manipulating attention layers (Stacey et al., 2022), designing loss metrics to penalize learning shortcuts (Rajič et al., 2022), and training other models to expose and/or correct spurious cues in the target model (Sanh et al., 2021; Karimi Mahabadi et al., 2020; Stacey et al., 2020). **Data-augmentation-based** approaches modify the dataset to mitigate spurious cues via data augmentation (Wu et al., 2022a; Lu et al., 2020; Nie et al., 2020).

Our proposed method is also based on data-augmentation: by introducing free-text explanations into the training data, we provide a signal for feature relevance which requires no prior knowledge of the spurious correlation. Concurrent to our work, Ross et al. (2022) also studied the impact of joint explain-and-predict training²⁸ for improving model robustness against spurious correlations. They find that the effect of the method scales positively with model size, which has similar results to our analysis of models in the GPT-3 family. In terms of the data used for training, they use two datasets known to contain artifacts, whereas we induce cues via *filtering* four different datasets (Section 7.3.1), which allows us to precisely control for the strength of each spurious correlation.

7.2. Problem Definition

The problem we want to solve is: given the training data containing some spurious correlation, how can we help the model overcome the correlation such that it better generalizes to out-of-distribution data?

Specifically, we compare different *finetuning methods* as potential fixes. Moreover, the finetuning methods should be agnostic to the cue. Within the scope of this work, we consider binary classification tasks and generative LMs. Following Kaushik et al. (2019), we select a set of spurious cues defined as features that correlate with, but do not causally influence, the label.

We construct the training and evaluation sets as follows: for each task T , we create a skewed training

²⁸Also known as "rationalization" or "self-rationalization" (Wiegrefe et al., 2021; Chen et al., 2022a).

set D_{train}^f , by intentionally introducing a spurious feature f into the training data, such that the presence of the cue perfectly correlates with one of the task labels; in addition, we have the unskewed training set D_{train} and test set D_{test} sampled from the original distribution, thus not containing the spurious correlation.

Now, our goal is to evaluate how a finetuning method FT affects a model’s robustness to the spurious correlation in D_{train}^f . In particular, we require FT to be agnostic to the feature f . Given a pretrained LM M , we first finetune it on the unskewed D_{train} using method FT , obtaining M_{base}^{FT} . We evaluate it on D_{test} , obtaining the base accuracy $acc(M_{base}^{FT})$. Then, we finetune M using method FT on the skewed D_{train}^f and evaluate the resulting model M_f^{FT} on D_{test} , obtaining its accuracy $acc(M_f^{FT})$. In addition, we compute the Matthews correlation coefficient (MCC)²⁹ between its predicted label and the feature f , denoted by $corr_f(M_f^{FT})$.

We measure the robustness of the model M_f^{FT} to the spurious cue f with the accuracy drop from the base level

$$\delta_{acc}^f(M, FT) := acc(M_f^{FT}) - acc(M_{base}^{FT})$$

and the prediction-feature correlation

$$corr_f(M_f^{FT}).$$

Let $M_f^{FT_1}$ and $M_f^{FT_2}$ be two models finetuned with methods FT_1 and FT_2 respectively. We say that $M_f^{FT_1}$ is more robust to feature f than $M_f^{FT_2}$ is if $\delta_{acc}^f(M, FT_1) > \delta_{acc}^f(M, FT_2)$ and $corr_f(M_f^{FT_1}) < corr_f(M_f^{FT_2})$. Our goal is to study how $\delta_{acc}^f(M, FT)$ and $corr_f(M_f^{FT})$ change with different finetuning methods FT , which we detail in the next section.

7.3. Method

With the above formalization, we now describe the process used to generate the skewed training set D_{train}^f for a spurious cue f and the different finetuning methods FT we consider.

²⁹Matthews correlation is commonly used to measure the association between two binary variables. It is the Pearson correlation in the binary setting.

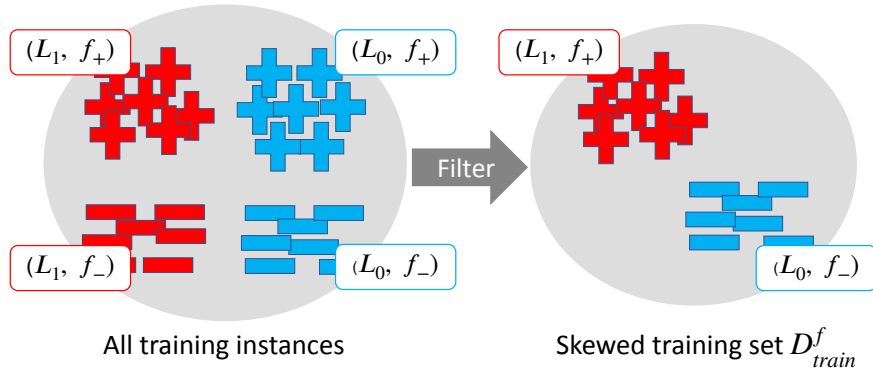


Figure 7.2: We filter the training data to introduce spurious correlations. The color represents the label, e.g. Offensive and Not offensive. The shape represents the presence of a feature, e.g. whether a post contains username mentions. The resulting D_{train}^f contains 500 examples of (L_1, f_+) and 500 examples of (L_0, f_-) .

7.3.1. Constructing Skewed Training Sets

We construct the skewed D_{train}^f via filtering. Consider a binary classification task T (e.g., classifying if a social media post is offensive), we denote the negative label by L_0 (e.g., Not offensive) and the positive label by L_1 (e.g., Offensive). We want to introduce a spurious feature f (e.g., username mentions) into the training data, such that its presence correlates with the label. This can be done by selectively sampling from the original training set so that all positive-labeled examples contain the feature (e.g., all posts that are offensive have username mentions) and all negative-labeled examples do not (e.g., all posts that are not offensive have no username mentions).

As shown in Figure 7.2, each resulting D_{train}^f contains 1,000 examples: 500 positive-labeled instances where the feature f is present (L_1, f_+) , and 500 negative-labeled instances which do not contain the feature (L_0, f_-) . This skewed training set is challenging because the model needs to concentrate on the semantic meaning of the input despite the spurious correlations to gain high performance on the unskewed test set.

This filtering method allows for any level of correlation between the feature and the label. For our main results in Section 7.5, we use skewed training sets with an MCC of 1.0 to evaluate performances in the worst case. In Section 7.6, we perform additional experiments varying the levels of correlation.

7.3.2. Finetuning Methods

We compare the two finetuning methods illustrated in Table 7.1. In **standard finetuning**, we feed the input text (e.g., “Does the premise ‘Children smiling and waving at camera’ entail the hypothesis ‘There are children present’?” from the e-SNLI dataset) to the model, and let it generate a binary label (True/False). In **explanation-based finetuning**, given the same input, the model additionally generates a free-text explanation (e.g., “The children must be present to in order to see them”) followed by the label.

7.4. Experimental Setup

7.4.1. Datasets

We consider four binary text classification tasks³⁰ with human-annotated free-text explanations, exemplified in Table 7.1:

CREAK (Onoe et al., 2021) Given a claim, the task is to verify whether it is True (L_1) or False (L_0).

e-SNLI (Camburu et al., 2018a) Given a premise and a hypothesis, the task is to decide whether it is True (L_1) or False (L_0) that the premise entails the hypothesis.³¹

ComVE (Wang et al., 2019) Given two sentences, the task is to judge which one of Sentence 1 (L_1) or Sentence 2 (L_0) is more plausible.

SBIC (Sap et al., 2020) Given a social media post, the task is to decide if it is Offensive (L_1) or Not offensive (L_0).

For each dataset, we sample 1,000 instances for the skewed training set D_{train}^f following the method presented in 7.3.1. Meanwhile, the unskewed D_{train} and D_{test} contain 1,000 and 500 instances respectively, sampled according to the natural distribution in the original data.

³⁰The last three datasets are from the FEB benchmark (Marasovic et al., 2022a).

³¹We convert the original 3-way classification to binary classification by considering both Neutral and Contradiction as non-entailment.

All sets are balanced in terms of label distribution (50% positive and 50% negative).

7.4.2. Spurious Cues

We introduce a diverse set of binary cues, including human-detectable cues, and cues that are not detectable by humans (e.g., embedding clusters).³² All these cues are spurious in the sense that their presence or absence does not causally influence the ground truth label.

Sentence Length. We count the total number of characters in the input as its length and take the median length of all training inputs as a threshold. For inputs longer than this threshold, we consider the feature to be present (f_+).

Present Tense. We perform tokenization and Part-of-Speech (POS) tagging on the input. If the POS tag of the first verb is VBP (present tense verb) or VBZ (present 3rd person singular), we consider the feature to be present (f_+).

Plural Noun. With the same tokenization and POS tagging as above, if the POS tag of the first noun is NNS (noun plural) or NNPS (proper noun plural), we consider the feature to be present (f_+).

Embedding Cluster. We use Sentence-BERT (Reimers and Gurevych, 2019) to generate embeddings for each input and run K-Means Clustering on the training set to assign inputs into two clusters, arbitrarily indexed as C_0 and C_1 . If an input falls in cluster C_0 , we consider the feature to be present (f_+). Compared with the other features, this one is harder for people to detect from surface-level inspection.

7.4.3. Evaluation Metrics

As discussed in Section 7.2, in order to evaluate the robustness of M_f^{FT} (the model finetuned with method FT) to the spurious feature f , we measure the accuracy drop $\delta_{acc}^f(M, FT)$ from the base level and the prediction-feature correlation $corr_f(M_f^{FT})$. A higher $\delta_{acc}^f(M, FT)$ (since it is typically negative) or a lower $corr_f(M_f^{FT})$ indicates higher robustness to the spurious correlation.

³²We also experiment with dataset-specific cues, described in Appendix D.3.

		ComVE		CREAK		e-SNLI		SBIC	
		Standard	Explain	Standard	Explain	Standard	Explain	Standard	Explain
Accuracy (δ_{acc})	No Spurious Cue	97.0	95.6	84.2	85.0	91.6	89.2	79.0	75.0
	Sentence Length	91.4 (-5.6)	89.4 (-6.2)	60.4 (-23.8)	80.2 (-4.8)	69.8 (-21.8)	76.2 (-13.0)	50.4 (-28.6)	53.4 (-21.4)
	Present Tense	93.6 (-3.4)	93.0 (-2.6)	74.6 (-9.6)	80.2 (-4.8)	76.0 (-15.6)	86.6 (-2.6)	78.6 (-0.4)	77.4 (2.4)
	Embedding Cluster	85.6 (-11.4)	89.8 (-5.8)	69.2 (-15.0)	78.6 (-6.4)	70.6 (-21.0)	89.2 (0.0)	70.6 (-8.4)	71.8 (-3.2)
	Plural Noun	96.8 (-0.2)	94.6 (-1.0)	72.2 (-12.0)	77.2 (-7.8)	69.0 (-22.6)	85.4 (-3.8)	74.0 (-5.0)	80.6 (5.6)
	Average	91.9 (-5.1)	91.7 (-3.9)	69.1 (-15.1)	79.1 (-6.0)	71.4 (-20.3)	84.4 (-4.9)	67.9 (-11.2)	70.4 (-4.7)
	Prediction- Feature Correlation	Sentence Length	0.134	0.108	0.847	0.325	0.467	0.291	0.770
	Present Tense	0.074	0.035	0.305	0.146	0.336	0.055	0.241	0.166
	Embedding Cluster	0.291	0.172	0.563	0.288	0.595	0.147	0.430	0.363
	Plural Noun	0.060	0.064	0.445	0.170	0.578	0.221	0.047	-0.050
	Average	0.140	0.095	0.540	0.232	0.494	0.179	0.363	0.161

Table 7.2: Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of GPT-3 (Davinci, 175B), finetuned with and without explanations.

7.4.4. LMs

We experiment with the following generative LMs: GPT-3 (base models of Davinci, Curie, Babbage, Ada) (Brown et al., 2020b), T5 (base) (Raffel et al., 2020), BART (base) (Lewis et al., 2020b), and OPT (1.3b) (Zhang et al., 2022b) ³³ to assess whether our method works for models of different sizes and families.

7.5. Results

To reemphasize our research question, we want to know: **can explanations make models less susceptible to spurious cues?** Table 7.2 shows the performance of GPT-3 (Davinci) finetuned with and without explanations on all four datasets. When the training set is not skewed by a spurious cue (row 1), adding explanations generally does not contribute to model performance. Compared to standard finetuning, explanation-based finetuning decreases the accuracy by 1-4 on ComVE, e-SNLI, and SBIC. In CREAK, the accuracy only increases by 0.8.

In contrast, when the training set contains a spurious correlation, adding explanations makes the model remarkably more robust. This is true across the vast majority of datasets and spurious cues, as reflected by the accuracy drop $\delta_{acc}^f(M, FT)$ and the prediction-feature correlation $corr_f(M_f^{FT})$.

³³See Appendix D.1 for implementation details.

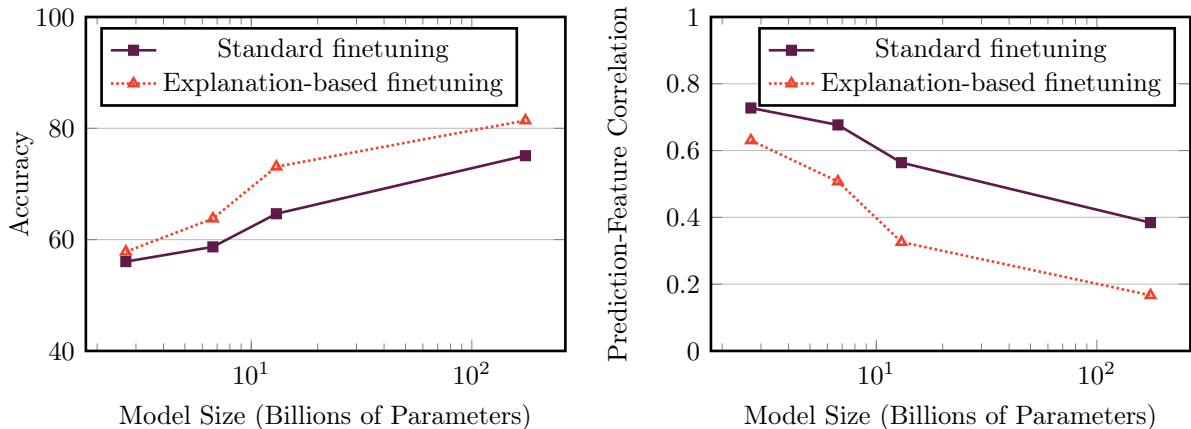


Figure 7.3: Accuracy (\uparrow) and prediction-feature correlation (\downarrow) across four GPT-3 models of different sizes (Ada 2.7B, Babbage 6.7B, Curie 13B, Davinci 175B). Accuracies and correlations are averaged across all five cues and all four datasets for each model.

Across all datasets, adding explanations in finetuning mitigates the average accuracy drop for models on the unskewed test set (by 1.2, 11.3, 15.4, and 6.5, respectively). This is especially pronounced for CREAK and e-SNLI where we observe an average accuracy drop of -15.1 and -20.3 respectively in standard finetuning, but only -3.8 and -4.9 in explanation-based finetuning.

Since adding explanations incurs a small accuracy penalty in the no cue condition, its benefits in terms of *absolute accuracy* is not always clear across all datasets. On ComVE, standard finetuning outperforms our method by 0.2. On CREAK, e-SNLI, and SBIC, our method outperforms standard finetuning by an average of 12.1, 13.0, and 2.5, respectively. Still, this represents an average accuracy gain of 6.9 across all datasets.

In terms of prediction-feature correlation, our method consistently results in a lower average correlation compared to standard finetuning (-0.045, -0.309, -0.315, and -0.202, on all datasets respectively). Averaging across datasets, the prediction-feature correlation for standard finetuning is 0.384, while for explanation-based finetuning it is only 0.167 (-0.217). This suggests that explanation-based finetuning makes models rely less on spurious cues.

Overall, there is strong evidence to support that including explanations during finetuning can make LLMs more robust to spurious correlations.

7.6. Analysis

Having shown the effectiveness of our method, we now analyze potential factors that may influence its performance by answering the following questions:

7.6.1. Do explanations improve the robustness of models of different sizes and families?

We run the experiments in Section 7.5 with three smaller GPT-3 models (Ada, Babbage and Curie), T5, BART and OPT. Full results for all models are given in Appendix D.2.

Figure 7.3 shows the results for the four GPT-3 models averaged across all cues and all datasets. Overall, explanations can still improve the robustness of all four models, though to a lesser extent for smaller ones. For GPT-3 Ada, for example, the absolute accuracy gain from explanation-based finetuning over standard finetuning averaged across all datasets and cues is 1.78, as opposed to 6.85 for Davinci. As for the average prediction-feature correlation, including explanations in finetuning reduces the correlation by 0.122 ($0.728 \rightarrow 0.606$) for Ada, which is smaller than the reduction for Davinci (0.217).

Interestingly, when no spurious cue is introduced, adding explanations substantially decreases smaller models’ accuracy across all datasets (e.g., by an average of 13.2 for Ada). For Davinci, this average drop is only 1.75, smaller but still non-zero. This suggests that it is more challenging for smaller models to generate good explanations, so the accuracy penalty from explanation-based finetuning is more severe. By contrast, larger models benefit more from our method. This is likely due to their capability of producing higher-quality explanations.

Observing the full results for all models from Appendix D.2, we see that our method lowers the prediction-feature correlation across all model families studied (GPT-3, OPT, BART, and T5) but only improves absolute accuracy for all four GPT-3 models and OPT. This may also be due to scale since the BART (110M) and T5 (220M) base models we experiment with are notably smaller than the OPT (1.3b) model and the smallest GPT-3 model (2.7b). Interestingly, while our method yields the greatest gains for Davinci (175B), Curie still experiences 95% of the accuracy gains we see in Davinci, despite being less than a tenth of Davinci’s size. These results suggest that our method can

be useful for other open-source models, many of which are in a similar size range.

7.6.2. How does the spurious correlation strength affect our method?

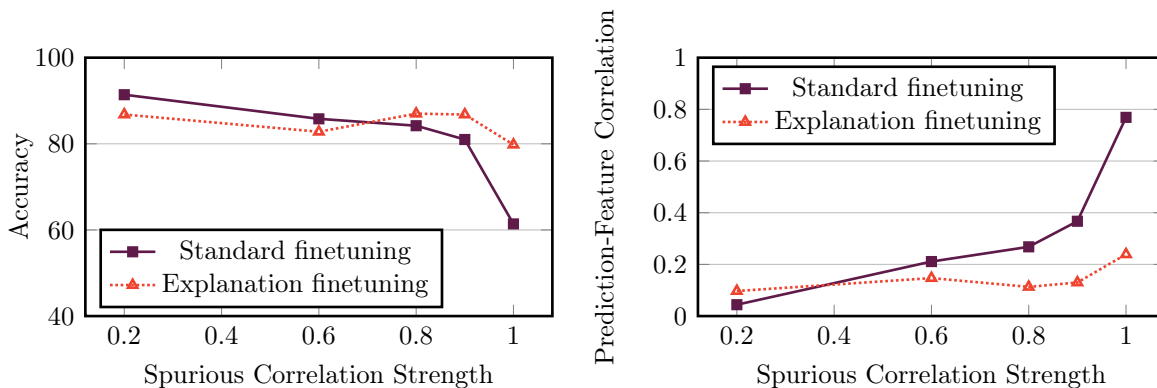


Figure 7.4: Accuracy (\uparrow) and prediction-feature correlation (\downarrow) of GPT-3 (Davinci) on e-SNLI, as the strength of the “embedding cluster” spurious correlation varies.

The strength of the spurious correlation in our skewed training set is maximum for the main experiments presented in Section 7.3.1. This means that the cue is perfectly correlated with the label (MCC=1.0). Here, we analyze how our method works with different levels of spurious correlation strength in the training set. We select e-SNLI and the embedding cluster cue as a case study. Note that in the main experiments with MCC=1.0, we only sample positive-labeled examples from the pool of examples with the feature present (L_1, f_+) and negative-labeled examples from examples with the feature absent (L_0, f_-). Here, we vary the level of correlation by introducing a certain number of negative-labeled examples containing the feature (L_0, f_+) and positive-labeled examples not containing the feature (L_1, f_-) into the training set.

As shown in Table 7.2, standard finetuning for e-SNLI outperforms explanation-based finetuning by 2.4 in terms of accuracy under the “no cue” condition, where the correlation between the label and the embedding cluster feature is near zero.

When the correlation becomes 1.0, this difference is 18.6 in favor of explanation-based finetuning. The “no cue” and perfect correlation conditions represent two extreme cases.

We show the results with different levels of spurious correlation strength in Figure 7.4, in terms of

		CREAK			e-SNLI		
		Standard	Explain	Permute	Standard	Explain	Permute
Accuracy (δ_{acc})	No Cue	84.2	85.0	86.2	91.6	89.2	90.0
	Sentence Length	60.4 (-23.8)	80.2 (-4.8)	67.6 (-18.6)	69.8 (-21.8)	76.2 (-13.0)	72.2 (-17.8)
	Present Tense	74.6 (-9.6)	80.2 (-4.8)	75.4 (-10.8)	85.8 (-5.8)	88.0 (-1.2)	80.2 (-9.8)
	Embedding Cluster	69.2 (-15.0)	78.6 (-6.4)	74.8 (-11.4)	70.6 (-21.0)	88.6 (-0.6)	77.4 (-12.6)
	Average	68.1 (-16.1)	79.7 (-5.3)	72.6 (-13.6)	75.4 (-16.2)	84.3 (-4.9)	76.6 (-13.4)
	Prediction- Feature Correlation	Sentence Length	0.847	0.325	0.457	0.467	0.291
	Present Tense	0.305	0.146	0.319	0.217	0.143	0.322
	Embedding Cluster	0.563	0.288	-0.427	0.595	0.141	-0.303
	Average	0.572	0.253	0.116	0.426	0.192	0.134

Table 7.3: Results on CREAK and e-SNLI when explanations are permuted to be completely irrelevant to the input, in comparison with standard finetuning and explanation-based finetuning (with valid explanations).

accuracy and prediction-feature correlation.

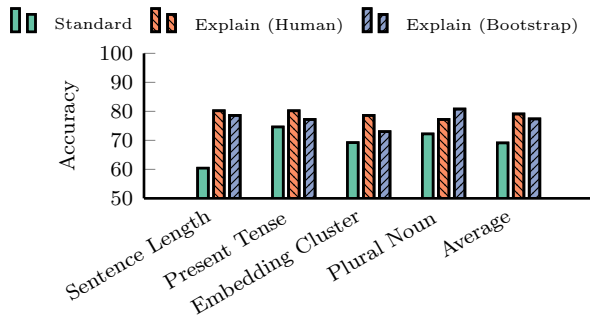
We observe that explanation-based finetuning starts to perform better than standard finetuning when the correlation between the spurious cue and the target feature is above 0.8.

7.6.3. Does explanation quality affect the effectiveness of our method?

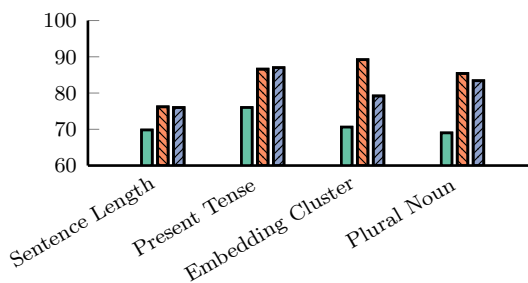
In the in-context learning scenario, Lampinen et al. (2022b) show that explanations can improve task performance when used in few-shot prompting. Specifically, they find that high-quality explanations that are manually selected provide substantially more gains than explanations that are not filtered for quality.

To analyze the impact of explanation quality in our setting, we intentionally lower the quality of explanations provided during finetuning by making them irrelevant to the input. We do this via *in-label permutation* on all explanations: for any given instance in the training set, the explanation for its label will be replaced with the explanation from another instance with the *same* label. In other words, the new explanation does not apparently conflict with the label but is irrelevant to the input.

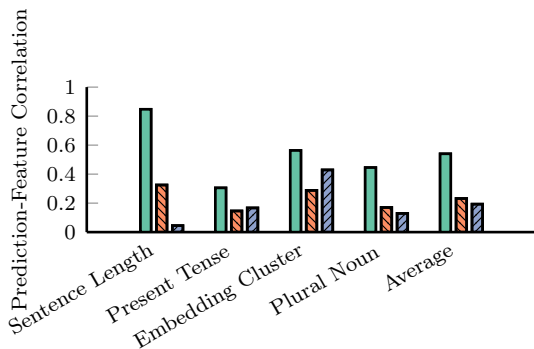
We experiment with datasets where explanation-based finetuning shows the largest benefits (CREAK and e-SNLI). The results are shown in Table 7.3. Surprisingly, even with permuted explanations,



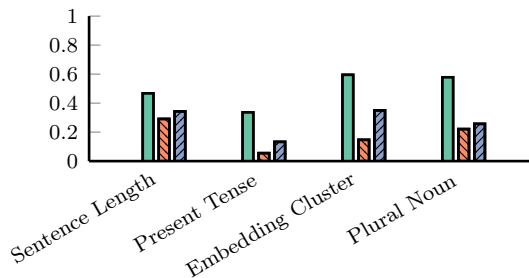
(a) Accuracy(\uparrow) on CREAK



(b) Accuracy(\uparrow) on e-SNLI



(c) Prediction-Feature Correlation(\downarrow) on CREAK



(d) Prediction-Feature Correlation(\downarrow) on e-SNLI

Figure 7.5: Results for finetuning with bootstrapped explanations (**Explain (Bootstrap)**), in comparison to finetuning without explanations (**Standard**) and finetuning with human-written explanations (**Explain (Human)**).

our method still provides a benefit over having no explanations at all. Averaging over all spurious cues and both datasets, the accuracy gain from using permuted explanations compared to having no explanations is 2.85. Naturally though, this is much smaller than the accuracy gain from using the non-permuted explanations (10.25).

These results can be compared with the findings from Wang et al. (2022a) which show the central role of explanation relevance in the few-shot setting. To understand why permuted explanations still help in our case, since our data contains spurious cues, we hypothesize that the model might be “distracted” by the explanations even if they are irrelevant, and could thus “forget” the spurious cues. We leave it for future work to verify this hypothesis.

7.6.4. Do the explanations have to be human-written?

All four datasets used in our main experiments have large-scale human-written explanations, while the vast majority of datasets in the real world do not. In this analysis, we investigate the possibility of using LM-generated explanations instead of human-written ones, to see if it is possible to generalize our method to datasets for which human explanations are not available.

We also use the CREAK and e-SNLI datasets in this experiment as a case study. We prompt GPT-3 (Davinci) in a 10-shot setting to generate an explanation for a given input. We do this via a bootstrapping process that starts with 10 labeled training instances which we then grow in an iterative fashion to add explanations to examples in the training set without explanations. The four step process is as follows: (1) we initialize the seed set with 10 training instances, including the label and the human-written explanation; (2) we sample 10 instances without replacement from the seed set, and prompt the model to generate an explanation for a new instance from the training set; (3) we add the new instance with the generated explanation to the seed set; (4) we repeat steps (2)-(3) until the entire training set contains explanations. Note that when generating the explanation, we give the model access to the ground-truth label. The temperature is set to 0.9 to facilitate diverse completions.

Results obtained with these explanations generated via bootstrapping are shown in Figure 7.5a and Figure 7.5c for CREAK and in Figure 7.5b and Figure 7.5d for e-SNLI. On average, finetuning with bootstrapped explanations results in an accuracy gain of 8.3 for CREAK and 10.1 for e-SNLI, compared to standard finetuning without any explanations. Although these improvements are slightly lower than those obtained with human-written explanations (10.0 for CREAK and 13.1 for e-SNLI), they are nevertheless substantial. Inspecting the prediction-feature correlation for CREAK, bootstrapped explanations induce an average correlation drop of 0.347 compared to standard finetuning, surprisingly surpassing the drop achieved with human-written explanations (0.308). In the case of e-SNLI, the prediction-feature correlation drops by an average of 0.223 for bootstrapped explanations which, despite not being as substantial as with human-crafted explanations (0.316), is still a significant improvement. These results indicate that explanation-based finetuning can be

beneficial for datasets without human-provided explanations, and illustrate the generalizability and applicability of our approach to more datasets.

7.7. Conclusion

In this chapter, we studied whether incorporating explanations as training signals can help reduce model reliance on spurious cues in the data, to demonstrate the utility of explanations during finetuning.

Specifically, in addition to predicting a label, models are finetuned to also generate a free-text explanation in support of its prediction. We perform experiments on a diverse set of classification tasks involving different types of spurious features. Results show that our method makes the models substantially more robust towards spurious features, as measured by both accuracy and correlation-based metrics. The efficacy of our method generalizes to different model sizes and families, though larger models tend to benefit more. Moreover, we observe that the stronger the spurious correlation in the data, the more helpful our method is. Interestingly, we show that highly relevant explanations are not absolutely necessary, since permuted explanations still provide around 25% of the accuracy benefits observed with non-permuted explanations. What is most notable is that even with model-generated explanations, our method works almost as well as with human-written ones, implying its potential applicability to the vast majority of datasets for which human-written explanations are not available.

We note a few key limitations of our approach. Similar to what was shown by previous interpretability studies (Camburu et al., 2018a, i.a.), incorporating explanations comes with some penalty on in-distribution accuracy when there is no spurious cue. This penalty decreases as model size increases, potentially because it is less challenging for larger models to generate good explanations. The second limitation is that our artificially constructed training set may not reflect the strength of the studied spurious cues in the real world. In our main experiments, we focus on the case where one spurious cue is perfectly correlated with the target label. For further exploration, we can study the alternative setting where there are multiple weak spurious cues instead of a single strong one. Finally, our work here is limited by the scope of the experiments. We only experiment with generative LMs and

binary classification tasks. Also, because of resource constraints, we only consider four datasets and eight types of spurious cues (including dataset-independent and dataset-specific ones). Additional experiments using a wider variety of spurious cues and datasets would help to shed light on how our method generalizes to other scenarios.

This chapter further solidifies the utility of explanations by focusing on their potential to improve robustness during the finetuning process. Previously, we have studied the utility of explanations during in-context learning, towards calibration (Chapter 5) and self-correction (Chapter 6). The findings from this chapter contribute to the broader theme of this thesis by showing that explanations are useful in helping mitigate the reliance of LLMs on spurious cues, thereby enhancing their robustness. This insight is particularly valuable for developing more reliable and trustworthy AI systems, which are less likely to be misled by irrelevant features in the data. The observation that model-generated explanations can be nearly as effective as human-written ones opens up possibilities for applying this method to a wide range of datasets without the need for extensive human annotation.

Looking ahead, the next chapter will conclude our exploration by reflecting on the cumulative findings of this thesis. We will discuss the key takeaways on enhancing the faithfulness and utility of LLM-generated explanations, and outline potential directions for future research.

CHAPTER 8

Conclusion and Future Research

8.1. Conclusion

This thesis set out to address the challenges in explanations generated by LLMs, with a focus on faithfulness and utility. We started by discussing the development of LLMs, the notion of explainability in NLP, and their intersection — LLM-generated explanations. We contextualized this line of work in a taxonomy of model explanation methods in NLP, as a type of “self-explanations”. We then identified two critical gaps in existing studies along this line – faithfulness (explanations should accurately reflect the model’s reasoning mechanism behind the prediction) and utility (explanations should be helpful towards achieving pre-defined goals in real-world applications). To address the first gap, we proposed a two-stage reasoning framework called Faithful CoT, integrating stochastic Translation and symbolic Problem Solving. To address the second gap, we demonstrated the practical utility of explanations during in-context learning and finetuning, towards enhancing LLMs’ capabilities in calibration, robustness, and potentially self-correction. Notably, we highlight several key takeaways:

LLM-generated explanations are not faithful or useful by default. LLMs possess an impressive ability to generate plausible-looking text after being trained on extensive datasets. These generated explanations often appear convincing, which can lead to a false sense of trust in their predictions. However, as discussed in Chapter 2.3, these explanations are not necessarily faithful. LLMs can “lie” about their reasoning, rationalizing their predictions with explanations that do not reflect the actual biases used in prediction. Furthermore, explanations are not trivially useful. When used without careful consideration, they can be either useless or even adversarially helpful, particularly in collaborative decision-making scenarios. Therefore, it is crucial not to blindly trust LLM-generated explanations as transparent windows into model behavior and to be cautious about claims of “self-interpretability”.

Integrating symbolic structures can enhance the faithfulness of LLM-generated expla-

nations. The auto-regressive nature of the generation process in LLMs means there is no intrinsic constraint linking the final prediction with the explanation. As a result, the reasoning described in the explanation may differ entirely from the process used to generate the prediction. To mitigate this issue, we introduced symbolic structures to constrain the prediction derivation process. By deterministically deriving the prediction through the execution of a symbolic reasoning chain, we ensure that the explanation faithfully represents how the answer is arrived at. This approach guarantees faithfulness and, as demonstrated through various reasoning tasks, also improves performance, showcasing the synergy between faithfulness and accuracy.

When used as an exposure interface or training signals, LLM-generated explanations can be helpful for model improvement in different scenarios. During in-context learning, explanations are helpful as an interface for the model to explicitly verbalize its thought process, allowing humans or the model itself to inspect and improve on its outputs. Prompting LLMs to produce explanations improves their calibration, with larger models benefiting more significantly. This method also holds potential for enhancing self-correction capabilities, although further improvements in critiquing and editing stages are necessary. During finetuning, incorporating explanations as training signals guides models towards relevant features, reducing reliance on spurious correlations in the dataset. This practice enhances robustness, as models forced to articulate explanations before making predictions are less likely to depend on irrelevant features.

While faithfulness sometimes enhances utility, this relationship is not straightforward. Intuitively, one might expect that faithful explanations would improve utility, but our findings indicate mixed evidence. In Chapter 5, we observed that faithful explanations improve calibration for instruction-tuned models, while unfaithful explanations are preferred for other models. In Chapter 6, different types of explanations showed similar levels of efficacy in self-correction. The factors determining the relationship between faithfulness and utility remain unclear, indicating a need for further research to understand this dynamic fully.

In conclusion, this thesis has made significant strides in addressing the challenges of faithfulness and utility in LLM-generated explanations. By proposing novel frameworks and demonstrating practical

applications, we have contributed to the growing body of knowledge in this field. However, our findings also underscore the complexity of these issues and the need for continued research to fully understand and leverage the potential of LLM-generated explanations in various applications.

For future work, I aim to further push the limits of the broader concept of Explainable AI (XAI), specifically in two directions: **controllability** and **human-centeredness**. The subsequent sections in this chapter will detail each direction.

8.2. Controllable Knowledge Adaptation with Symbolic Representation

In conventional end-to-end LM workflows, we frequently encounter challenges such as (a) how to determine when to utilize parametric knowledge versus in-context knowledge (like that provided by Retrieval-Augmented Generation or RAG), particularly when conflicts arise between the two; (b) how to generalize to new Domain-Specific Languages (DSL) or unfamiliar tools; and (c) how to unlearn sensitive or undesired information that has been accidentally learned by the model.

In Chapter 4, we introduced Faithful CoT, a method that decomposes complex tasks into Translation (with end-to-end LMs) and Problem Solving (with symbolic reasoners). While this approach proved effective, our experiments were limited to one-off generation without ongoing interaction.

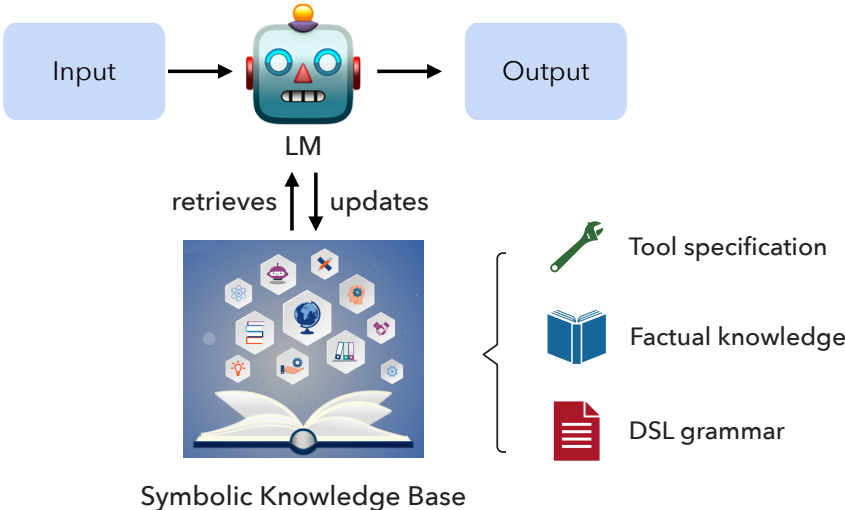


Figure 8.1: A illustration of an end-to-end LM integrated with an external symbolic “Knowledge Base” (KB), which can contain factual knowledge, Domain-Specific Language (DSL) grammars, tool specifications, and so on.

Building upon this foundation, a broader notion is to decouple knowledge and reasoning in LMs. This decoupling can be achieved by integrating a generalized symbolic “Knowledge Base” (KB) with the LM, as shown in Figure 8.1. This KB can encompass factual knowledge, DSL grammars, tool specifications, and more. By allowing the LM to retrieve from and update this KB, we can address the aforementioned challenges more effectively. For example, when new information is introduced in context, we can explicitly check for and resolve conflicts with existing facts in the KB. Meanwhile, transitioning between different DSLs becomes a matter of swapping the KB from one set of grammars to another. Finally, the removal of sensitive information can be accomplished by simple deletion of the relevant facts in the KB.

By explicitly materializing and controlling LM knowledge through a symbolic external KB, we can enhance the reliability and adaptability of XAI systems.

8.3. Human-Centered Evaluation and Application of XAI Systems

In Chapter 5 through 7, we demonstrated the utility of explanations for model improvement during finetuning and in-context learning stages, enhancing model robustness and calibration. Still, the application of XAI systems in real-world scenarios involving human interaction, such as model debugging, auditing, or collaborative decision-making, remains under-explored. While XAI promises to be beneficial in these contexts, real-world implementations often fall short of expectations. For instance, Bansal et al. (2021) found in a user study that the mere presence of an explanation, regardless of its content, can result in over-reliance on AI decisions. This highlights a critical gap between the theoretical potential of XAI and its practical efficacy. To address this discrepancy, we argue that the design and evaluation of XAI systems should be grounded in specific use cases. This necessitates a human-centered evaluation framework comprising two key components:

- **More benchmarks for utility evaluation** across *diverse use cases*, including debugging, auditing, collaborative decision-making, knowledge discovery, and so on. While real user feedback is ideal, its lack of scalability presents challenges. Alternative methods, such as simulated user feedback in Chen et al. (2023a), may offer a viable solution.

- **Standardized evaluation constructs** that go beyond assessing mere *plausibility* (how convincing the explanation is to humans) to measure *utility* (how useful the explanation is for the target goal). For instance, in collaborative decision-making scenarios, metrics such as decision accuracy, speed, and the rate of decision changes could provide valuable insights.

Furthermore, the application of XAI systems should prioritize **generalization**. It is important to assess the system’s utility across varied populations, locations, and settings, and evaluate its capacity to provide insights for both observed and counterfactual scenarios. By adopting this human-centered approach to both evaluation and application, we can bridge the gap between XAI’s theoretical promise and its practical impact, fostering the development of truly useful and reliable XAI systems.

APPENDIX A

APPENDIX FOR CHAPTER 4

A.1. Empirical Comparison with Concurrent Work

Two pieces of concurrent work, Program of Thoughts (PoT) (Chen et al., 2022b) and Program-Aided Language Models (PAL) (Gao et al., 2022), were announced on arXiv within three months of our work. Essentially, they both generate Python programs, or SL-only reasoning chains, to derive the answer. Our approach differs from them mainly in the additional component of structured NL comments, which decomposes the original problem into simpler, inter-dependent subproblems.

We perform an empirical performance comparison with them on the same set of 10 datasets used in our main evaluation. Since both papers have only tackled math reasoning and symbolic reasoning tasks, we reimplement their methods by using the “noNL” prompt in our ablation study from Section 4.4. The comparison is done with `code-davinci-002` as the underlying LM and greedy decoding.

As shown in Figure A.1, on 6 of the 10 datasets (including most MWP datasets, SayCan, and Date Understanding), PAL/PoT and Faithful CoT have very close accuracy (<2.0 difference). On AQUA, PAL/PoT is visibly better. On the remaining three datasets (StrategyQA, Sports Understanding, and CLUTRR), Faithful CoT reasonably outperforms PAL/PoT. This may suggest that our method has an advantage when the task requires extensive external knowledge (e.g., StrategyQA and Sports Understanding) or when the SL is not frequent in the LM’s pretraining data (e.g., Datalog, or our self-defined relational expressions).

Finally, note that the key contribution of our method lies in interpretability. Though the addition of structured NL comments sometimes does not make a difference in performance, it does make the reasoning chain more understandable to the user. Furthermore, it may even enable users without a programming background to debug the model, by only interacting with the NL subproblems (e.g., adding/removing/editing a subproblem), which is worth further exploration in the future.

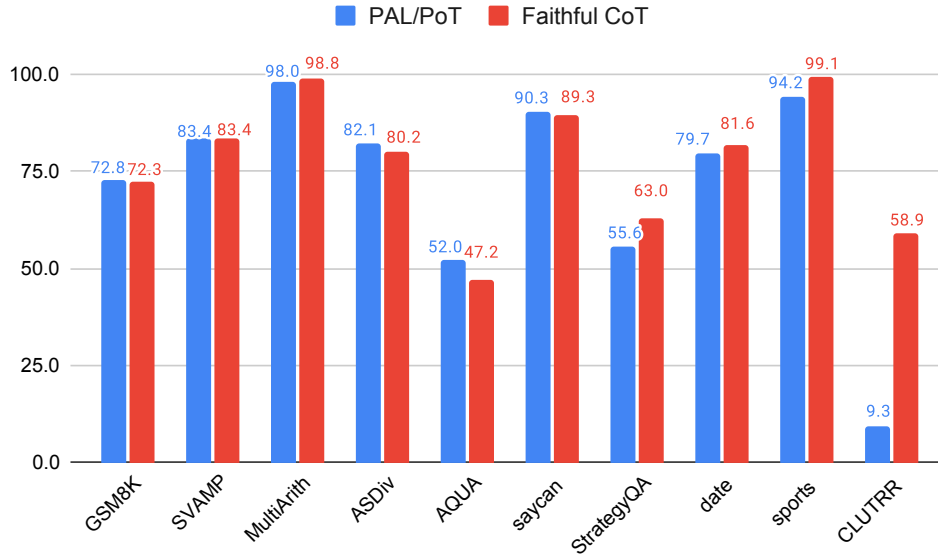


Figure A.1: Accuracy of our method and two concurrent methods, Program of Thoughts (POT) (Chen et al., 2022b) and Program-Aided Language Models (PAL) (Gao et al., 2022), on 10 reasoning datasets.

A.2. Dataset Details

A.2.1. Statistics

We show the dataset details in Table A.1, including the statistics, the number of few-shot exemplars used in the prompt, and example inputs and outputs.

In particular, we notice that in one of our baselines Wei et al. (2022b), the reported number of exemplars used in the prompt is inconsistent between the main text (10) and the appendix (6). To ensure fair comparison, we rerun the baseline with 10 exemplars for our results in Table 4.1, which is what we use for our method.

A.2.2. URLs and Licenses

We use the same distribution of datasets following Wei et al. (2022b):

Math Word Problems

- GSM8K (Cobbe et al., 2021): <https://github.com/openai/grade-school-math>, MIT license: <https://github.com/openai/grade-school-math/blob/master/LICENSE>.

Domain	Dataset	# Shot	# Test	Example
Math Word Problems	GSM8K	8	1,319	Q: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May? A: 72
	SVAMP	8	1,000	Q: Each pack of dvds costs 76 dollars. If there is a discount of 25 dollars on each pack. How much do you have to pay to buy each pack? A: 51
	MultiArith	8	600	Q: For Halloween Debby and her sister combined the candy they received. Debby had 32 pieces of candy while her sister had 42. If they ate 35 pieces the first night, how many pieces do they have left? A: 39
	ASDiv	8	2,096	Q: Seven red apples and two green apples are in the basket. How many apples are in the basket? A: 9
	AQuA	8	254	Q: A car finishes a journey in 20 hours at the speed of 60 km/hr. If the same distance is to be covered in 10 hours, how much speed does the car gain? A: "120 kmph"
Multi-hop QA	StrategyQA	6	2,290	Q: Did Aristotle use a laptop? A: False
	Date Understanding	10	359	Q: Yesterday was April 30, 2021. What is the date tomorrow in MM/DD/YYYY? A: "05/02/2021"
	Sports Understanding	10	977	Q: Is the following sentence plausible: "Lebron James hit the turnaround jumper"? A: True
Planning	SayCan	7	103	Q: Could you get me a drink with caffeine? A: "1.find(redbull) 2.pick(redbull) 3.find(user) 4.put(redbull) 5.done()."
Logical Inference	CLUTRR	8	1,042	Q: [Carlos] is [Clarence]'s brother. [Carlos] and his sister, [Annie], went shopping. [Annie] asked her mom [Valerie] if she wanted anything, but [Valerie] said no. How is [Valerie] related to [Clarence]? A: "mother"

Table A.1: Datasets used for evaluation. “# Shot” stands for the number of few-shot examples in the prompt (following Wei et al. (2022b)) and “# Test” stands for the number of test examples.

- SVAMP (Patel et al., 2021): <https://github.com/arkilpatel/SVAMP>, MIT license: <https://github.com/arkilpatel/SVAMP/blob/main/LICENSE>.
- MultiArith (Roy and Roth, 2015), license: CC BY 4.0.
- ASDiv (Miao et al., 2020): <https://github.com/chaochun/nlu-asdiv-dataset>.
- AQuA (Ling et al., 2017): <https://github.com/deepmind/AQuA>, license: <https://github.com/deepmind/AQuA/blob/master/LICENSE>.

Multi-hop QA

- StrategyQA (Geva et al., 2021): we use the open-domain setting (question-only set) from (BIG-Bench collaboration, 2021): https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/strategyqa.
- Date Understanding and Sports Understanding from BIG-Bench (BIG-Bench collaboration, 2021): Apache License v.2: <https://github.com/google/BIG-bench/blob/main/LICENSE>.

Planning

- SayCan (Ahn et al., 2022): SayCan dataset can be accessed at <https://say-can.github.io/> under CC BY 4.0 license.

Relational Reasoning

- CLUTRR (Sinha et al., 2019): <https://github.com/facebookresearch/clutrr>, license: <https://github.com/facebookresearch/clutrr/blob/main/LICENSE>. We obtain the publicly distributed version available at https://drive.google.com/file/d/1SEq_e1IVCDDzsBIBhoUQ5pOVH5kxRoZF/view, specifically the `data_089907f8` split.

We use all the above datasets for research purposes only, consistent with their intended use.

A.2.3. Data Cleaning

We perform manual cleaning on ASDiv, Date Understanding, Sports Understanding, and SayCan as we discover a number of annotation issues. In our experiment, we rerun all baselines on our cleaned version of the test sets. They are provided in our repository to assist future research.

Specifically, we clean each of the datasets as follows:

ASDiv: We start with the test set used by Wei et al. (2022b), which removes all questions with float-valued and string-valued answers. However, in their released version, we notice an error in the answer extraction step for questions with more than one value in the answer (e.g., “what is the width and length of X?”, where the answer consists of two values). In their implementation, only the first value is extracted as the ground truth answer, which is then compared against model outputs. This

might artificially inflate the final accuracy. To fix this, we extract all values in the answer as a set and compare model outputs against it.

Date Understanding: We find a number of wrong answers in the test set. For example, for the question “Jane and John married on Jan 2, 1958. It is their 5-year anniversary today. What is the date today in MM/DD/YYYY?”, the provided answer is “01/02/1961”, whereas the correct answer should be “01/02/1963”. We manually correct these answers, and the resulting test set has the same number of examples as the original one.

Sports Understanding: We notice a few ambiguities with the Sports Understanding dataset. For instance, running out of bounds is illegal in many sports. The phrase "Domantas Sabonis ran out of bounds" is labeled as implausible, however, Domantas Sabonis is a basketball player, and basketball players can indeed run out of bounds on the court. We remove 8 questions with such action-based ambiguities. Additionally, since the release of this dataset, a few new athletes have risen to fame with identical names to those mentioned in the dataset. For example, the question "Chris Paul struck out the side" is implausible, as the referenced “Chris Paul” is a famous basketball player. However, “Chris Paul” is also the name of a new MLB baseball player, in which case this statement is plausible. We remove 5 questions with such name-based ambiguities.

SayCan: We discover a few issues in the test set: (1) the environment setup (e.g., the list of objects, the list of locations, and the initial location of each object) is not the same for all examples; (2) the annotation of the ground truth answer is often incomplete (i.e., for a given task like “visit all locations”, there exist many possible plans in terms of the order of locations visited, but not all of them are included in the annotation); (3) there are ambiguous descriptions in certain queries, for example, in “Could you get me something refreshing?”, it is unclear what drinks are considered “refreshing”. For these questions, we complete the annotation whenever possible, and filter out the rest. The resulting test set contains 103 examples out of the original 120.

A.2.4. Dataset Splits

As stated in Section 4.2, we use the official splits whenever possible: training set for exemplar selection, validation set for prompt tuning, and test set for evaluation. In cases where they are available, we adopt the following strategies for each dataset:

GSM8K: it only has training and test sets. We form the validation set by randomly sampling 1,000 examples from the training set.

Other MWP datasets: for AQuA, we use the official training/validation/test split. For the other datasets, only the test sets are used, since we have the same prompt for GSM8K and them.

Date Understanding and **Sports Understanding**: they only have test sets. We follow Wei et al. (2022b) to select the same number of examples from the test set to form the few-shot prompt and use the remaining examples as a new test set.

SayCan: Following Wei et al. (2022b), we manually write 7 few-shot exemplars, since no training set is provided. We evaluate the models on our cleaned version of the test set, described in the previous subsection.

CLUTRR: this dataset is split into multiple folds. There is a training fold with $K \in \{2, 3\}$ (where K is the number of intermediate steps required to reach the answer), and one test fold for each K from 2 to 10. We construct the few-shot prompt using exemplars from the training fold, and test our method on the concatenation of all test folds.

A.3. Error Analysis

To further investigate where our method still fails, we inspect 100 errors³⁴ from model predictions on each of the four datasets and manually annotate the error categories.

³⁴To encourage sample diversity, we embed all the errors using `text-davinci-002` and cluster the embeddings using spectral clustering. This produces around 70 clusters of different sizes, from which we gather 100 samples using importance sampling.

A.3.1. StrategyQA

As mentioned in Section 4.3, Figure A.2 shows the error type distribution on a sample of 70 instances from StrategyQA, where we specifically compare the cases where the prediction of CoT is correct whereas ours is wrong.

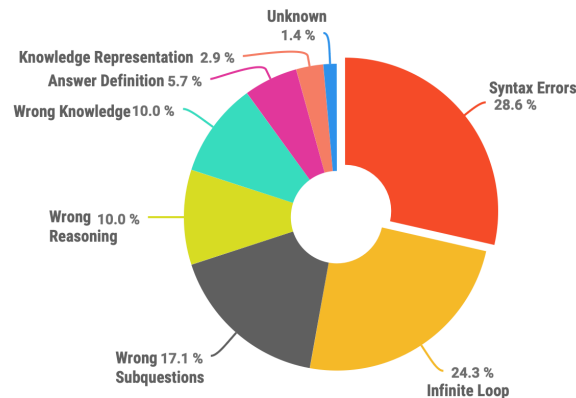


Figure A.2: Error analysis for StrategyQA.

There are 8 error types, from most to least frequent:

Syntax (29%): these are syntax-related errors that result in an invalid Datalog program. For example, the LM redefines a relation that is previously declared, uses an invalid string comparison operation, generates an ungrounded variable in the answer definition, and so on. With further pretraining on Datalog, we believe Codex is able to overcome these errors to a reasonable extent.

Infinite Loop (24%): the LM generates an infinite loop of subquestions, which are oftentimes rather philosophical. For example, for the question “Are potatoes native to the European continent?”, the LM generates a list of 40 subquestions (potentially more, if not truncated by the token limit):

1. Where are potatoes native to?
Potatoes are native to the Andes.
2. Which continent is the Andes in?
The Andes is in South America.
3. Which continent is South America in?

South America is in the Americas.

4. Which continent is the Americas in?

The Americas is in the Western Hemisphere.

5. Which hemisphere is the Western Hemisphere in?

The Western Hemisphere is in the Earth.

6. Which continent is the Earth in?

The Earth is in the Solar System.

.....

11. Which continent is the Universe in?

The Universe is in the Multiverse.

12. Which continent is the Multiverse in?

The Multiverse is in the Omniverse.

13. Which continent is the Omniverse in?

The Omniverse is in the Metaverse.

14. Which continent is the Metaverse in?

The Metaverse is in the Hyperverse.

.....

39. Which continent is the Quattuordecaverse in?

The Quattuordecaverse is in the Quindecaverse.

40. Which continent is the Quindecaverse in?

We notice that this type of errors also happen if the prompt does not include Datalog code but just the NL subquestions, but much less often.

Wrong Subquestions (17%): The LM generates an irrelevant subquestion to the original question. For instance, given the question, “Would toast for a vegan have margarine instead of butter?”, one subquestion that is generated is “What is the difference between a vegan and a vegetarian?”, which does not help with reaching the final answer.

Wrong Reasoning (10%): The subquestions are correct, but the reasoning from the retrieved facts

to the final answer is problematic. For example, to answer the question, “Do solo pianists require a conductor?”, the model correctly generates the relevant subquestions and their respective answers:

1. What is the definition of a solo pianist?

A solo pianist is a pianist who performs alone.

2. What is the definition of a conductor?

A conductor is a person who leads an orchestra.

3. What is the definition of an orchestra?

An orchestra is a large instrumental ensemble that contains string, brass, woodwind, and percussion instruments.

But in deriving the final answer, the reasoning is wrong: “Now, we derive the final answer: Do solo pianists require a conductor? The answer is Yes only if a solo pianist is not a conductor.”

Wrong Knowledge (10%): the LM fails to retrieve the correct knowledge to answer the subquestions. For example, given the original question “Is the largest city in New Mexico also known as Yootó?”, the model correctly generates the subquestions “What is the largest city in New Mexico?” (answer: Albuquerque) and “Is Albuquerque also known as Yootó?”. However, when answering the second subquestion, it retrieves a wrong piece of knowledge (“Albuquerque is also known as Yootó.”, whereas in reality, it should be “Santa Fe” that is known as Yootó).

Answer Definition (6%): In our prompt, we always derive the answer in the format of “The answer is Yes only if ...”, which is followed by a Datalog rule containing conditions that should be satisfied for the answer to be true. However, the LM sometimes generates this as “The answer is No only if ...”, which outputs the reversed answer.

Knowledge Representation (3%): The retrieved knowledge in NL is correct, but the representation of it in Datalog is wrong. For example, for the piece of knowledge “The Lucy Show is not the same TV series as JAG (TV series)”, the model represents it as follows:

```
.decl Same_TV_series(TV_series1:symbol, TV_series2:symbol)
```

```
Same_TV_series("The Lucy Show", "JAG (TV series)")."
```

which actually means the reverse (they are the same).

Unknown (1%): There is a very small proportion of errors (1 out of 70) where we are unsure of the cause. Specifically, we expect the solver to output True, but it outputs False instead.

A.3.2. Date Understanding

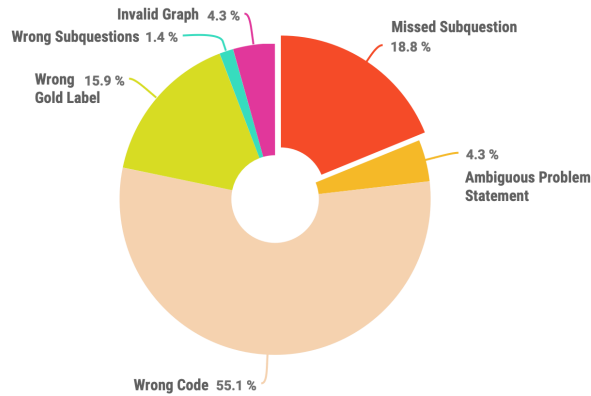


Figure A.3: Error analysis for Date Understanding. For a detailed description of the error categories, see Section A.3.2.

Unlike GSM8K, we only have 69 errors out of the 359 test examples, so we annotate them all, as shown in Figure A.3. The error categories for date understanding are similar to GSM8K, except that we do not see any generation errors in the samples, but we see questions with ambiguous phrasing allowing both the gold and predicted answers to be correct based on interpretation.

A.3.3. SayCan

Since SayCan only has 120 test examples and Faithful CoT produces 7 errors, we annotate all 7 of them, as shown in Figure A.4. These 7 examples can be categorized into two types:

Additional Subgoals (64%): Cases where the model generated unnecessary subgoals in the decomposition of the original task, leading the planner astray. This is illustrated by the request “Clear the jalapeno chips off the counter”:

```
(:goal  
  (and
```

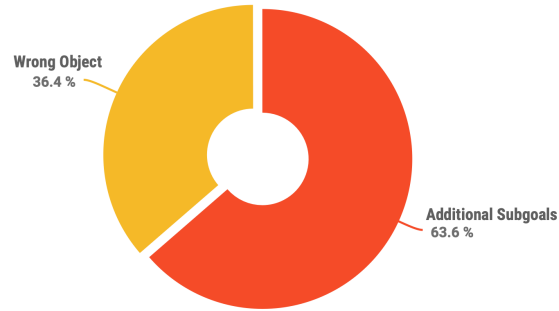


Figure A.4: Error analysis for SayCan. For a detailed description of the error categories, see Section A.3.3.

```

        (not (at jalapeno-chips counter))
        (not (at jalapeno-chips table))
        (not (at jalapeno-chips trash))
        (not (at jalapeno-chips bowl))
        (not (at jalapeno-chips user))
    )
)

```

Wrong Object (36%): Here the model generates the wrong object/object types in the goal. For example, a request such as “I opened a pepsi earlier. How would you bring me an open can?” fails because the model generates actions with water instead of Pepsi.

A.3.4. CLUTRR

For CLUTRR, we group all error cases by K , the number of steps in their gold reasoning chain, as a proxy for problem complexity, and perform importance sampling on these groups to select 100 examples. Our annotation of these examples reveals 5 error categories, as shown in Figure A.5:

Inversed Relation (41%): This stands out as the majority of the errors. These errors are caused by the reversal of directional relationships for the actors in the problem, i.e., predicting “mother” or “nephew” when the answer is “daughter” or “uncle” respectively.

Wrong Relation (30%): Here the model extracts the relation incorrectly (not even the inverse). For example, for the subquestion “How is [Donald] related to [Jason]?” with the correctly identified support “[Jason] is father of their father”, the model produces `relation(Donald, Jason) = son`

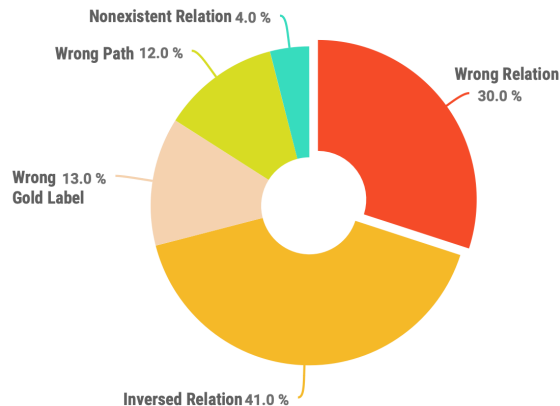


Figure A.5: Error analysis for CLUTRR. For a detailed description of the error categories, see Section A.3.4.

when the correct relation should be “grandson”.

Nonexistent Relation (4%): The model hallucinates a non-existent relation (e.g. “adopted” for daughter).

Wrong Path (12%): Here, the model does not generate a correct reasoning path from target entity A to target entity B in the question.

Wrong Gold Label (13%): These are annotation errors in the CLUTRR dataset. In one example, for the sentence, “[Gloria] asked her mother [Laura] if she could go outside and play with her friends.”, the annotation says Laura is Gloria’s grandmother.

A.4. Prompts

Due to the space limit, we show one exemplar in the prompt for each dataset here. The full prompts can be found in our repository.

Among all the MWP datasets, our prompt for AQuA is different from the rest, because the answers are in a multiple-choice format instead of integers. To produce a multiple-choice answer, we take a two-step approach by first producing a numerical answer in the same way as for the other math datasets. Then, we perform an additional step of converting the numerical answer into an answer choice by again prompting the language model to generate which answer choice is closest to the previously produced numerical answer. An exemplar of this 2-step prompt is shown in Table A.2.

EXEMPLAR FOR AQUA**Step 1: Answer Prediction**

```
# Question: In a flight of 600 km, an aircraft was slowed down due to bad weather. Its average speed for the trip was reduced by 200 km/hr and the time of flight increased by 30 minutes. The duration of the flight is:
# Answer option: ['A)1 hour', 'B)2 hours', 'C)3 hours', 'D)4 hours', 'E)5 hours']
# Write Python code to solve the following questions. Store your result as a variable named 'answer'.
# 1. What was the duration of the flight? (independent, support: ["The duration of the flight is"])
duration = Symbol('duration', positive=True)
# 2. What is the delay of the flight? (independent, support: ["the time of flight increased by 30 minutes"])
delay = 30 / 60
# 3. What was the total flight distance? (independent, support: ["In a flight of 600 km"])
total_distance = 600
# 4. What was the original speed? (depends on 1 and 3, support: ["External knowledge: speed is distance over time"])
original_speed = total_distance / duration
# 5. What was the reduced speed? (depends on 1, 2, and 3, support: [])
reduced_speed = total_distance / (duration + delay)
# 6. What was the duration of the flight if the original speed was 200 km/hr faster than the reduced speed? (depends on 4, 5, and 1, support: [])
solution = solve_it(original_speed - reduced_speed - 200, duration)
answer = solution[duration]
```

Step 2: Multiple Choice Conversion

```
# Question: In a flight of 600 km, an aircraft was slowed down due to bad weather. Its average speed for the trip was reduced by 200 km/hr and the time of flight increased by 30 minutes. The duration of the flight is:
# Answer option: ['A)1 hour', 'B)2 hours', 'C)3 hours', 'D)4 hours', 'E)5 hours']
# Prediction: 1.0000000000000000
# Closest Option: A
```

Table A.2: An exemplar from our prompt for AQUA.

EXEMPLAR FOR GSM8K, SVAMP, MULTIARITH, AND ASDIV

```
# Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
# To answer this question, write a Python program to answer the following subquestions:
# 1. How many trees are there in the beginning? (independent, support: ["There are 15 trees"])
trees_begin = 15
# 2. How many trees are there in the end? (independent, support: ["there will be 21 trees"])
trees_end = 21
# 3. How many trees did the grove workers plant today? (depends on 1 and 2, support: [])
trees_today = trees_end - trees_begin
# 4. Final Answer: How many trees did the grove workers plant today? (depends on 3, support: [])
answer = trees_today
```

Table A.3: An exemplar from our prompt for GSM8K, SVAMP, MultiArith, and ASDiv.

EXEMPLAR FOR STRATEGYQA

```
// Q: Would a pear sink in water?
// To answer this question, we answer the following subquestions:
// 1. What is the density of a pear?
// The density of a pear is about 0.6g/cm3.
// 2. What is the density of water?
// Water has a density of 1g/cm3.

// Then, we represent these answers in Datalog:
// 1. The density of a pear is about 0.6g/cm3.
.decl Has_density(Object:symbol, Density:float)
Has_density("pear", 0.6).
// 2. Water has a density of 1g/cm3.
Has_density("water", 1).

// Now, we derive the final answer: Would a pear sink in water?
// The answer is Yes only if the density of a pear is more than the density of water.
.decl Answer()
Answer() :- Has_density("pear", density1), Has_density("water", density2), density1 > density2.
.output Answer
```

Table A.4: An exemplar from our prompt for StrategyQA.

EXEMPLAR FOR DATE UNDERSTANDING

```
# Q: Yesterday was April 30, 2021. What is the date tomorrow in MM/DD/YYYY?
# To answer this question, we write a program to answer the following subquestions:
# import relevant packages
from datetime import date, time, datetime
from dateutil.relativedelta import relativedelta
# 1. What is the date yesterday? (independent, support: ["Yesterday was April 30, 2021"])
date_yesterday = date(2021,4,30)
# 2. What is the date today? (depends on 1, support: ["Yesterday was April 30, 2021"])
date_today = date_yesterday + relativedelta(days=1)
# 3. What is the date tomorrow? (depends on 2, support: [])
date_tomorrow = date_today + relativedelta(days=1)
# 4. Final Answer: What is the date tomorrow in MM/DD/YYYY? (depends on 3, support: [])
answer = date_tomorrow.strftime("%m/%d/%Y")
```

Table A.5: An exemplar from our prompt for Date Understanding.

EXEMPLAR FOR SPORTS UNDERSTANDING

```
# Q: Is the following statement plausible? Sam Darnold passed the puck
# To answer this question, write a Python program to answer the following subquestions:
# 1. Sam Darnold is a player in which sport? (independent, support: ["Sam Darnold is an NFL Quarterback",
"NFL is the National Football League"])
player_sport = "football"
# 2. The phrase "passed the puck" implies playing which sport? (independent, support: ["Players pass the
puck in hockey"])
playing_sport = "hockey"
# 3. Is the following statement plausible? Sam Darnold passed the puck (depends on 1 and 2, support:
["Sam Darnold is an NFL Quarterback", "NFL is the National Football League", "Players pass the puck in
hockey"])
plausibility = (player_sport == playing_sport)
# 4. Is the following statement plausible? Sam Darnold passed the puck (depends on 3, support: [])
answer = int(plausibility)
```

Table A.6: An exemplar from our prompt for Sports Understanding.

EXEMPLAR FOR SAYCAN

User query: Bring me something not sweet to eat.

Goal in PDDL:

```
(:goal
  ; I need to find a snack
  (exists (?s - snack)
    ; it has to satisfy the following conditions
    (and
      ; the snack must not be sweet
      (not (is-sweet ?s))
      ; bring it to the user
      (at ?s user)
    )
  )
)
```

Table A.7: An exemplar from our prompt for SayCan.

EXEMPLAR FOR CLUTRR

Context: [Jason] always had some great adventure planned for his granddaughter [Guillermina] when she came to visit. So, naturally, when [Myrna] told her daughter [Guillermina] that they would be going to visit [Jason] she could hardly contain herself.

Question: How is [Jason] related to [Myrna]?

To answer this question, we write a program to answer the following subquestions:

1. How is [Jason] related to [Guillermina]? (independent, support: "[Jason] always had some great adventure planned for his granddaughter [Guillermina] when she came to visit.")

relation Jason, Guillermina = grandfather

2. How is [Guillermina] related to [Myrna]? (independent, support: "So, naturally, when [Myrna] told her daughter [Guillermina] that they would be going to visit [Jason] she could hardly contain herself.")

relation Guillermina, Myrna = daughter

3. Final answer: How is [Jason] related to [Myrna]? (depends on 1, 2)

relation Jason, Myrna = relation Jason, Guillermina @ relation Guillermina, Myrna

Table A.8: An exemplar from our prompt for CLUTRR.

APPENDIX B

APPENDIX FOR CHAPTER 5

B.1. Implementation Details

B.1.1. Closed-Source Models

We use OpenAI Codex (`code-davinci-002`, deprecated since Jan 4, 2024) (Chen et al., 2021), GPT-3.5-turbo (`gpt-3.5-turbo-16k-0613`), and GPT-4 (`gpt-4-0613`) (OpenAI, 2023) through the Python API available at `platform.openai.com`, from Oct, 2023 to Feb, 2024. The inference cost per input query (with 40 samples of all five prompting strategies) is \$0 for all Codex models through the researcher access program, \$0.08 - \$0.13 for GPT-3.5-turbo, and \$0.61 - \$0.99 for GPT-4, depending on the dataset. The total cost of running inference on all 9 datasets is \$0 for Codex, around \$1,059 for GPT-3.5-turbo, and around \$7,942 for GPT-4. The inference time on one input query (with 40 samples of all five prompting strategies) is 50 - 95 seconds with Codex under a rate limit of 150,000 tokens/minute, 39 - 74 seconds with GPT-3.5-turbo under 2,000,000 tokens/minute, and 83 - 157 seconds with GPT-4 under 300,000 tokens/minute, also depending on the dataset. The total time for running inference on all 9 datasets is 8.3 days for Codex, 6.4 days for GPT-3.5-turbo, and 13.8 days for GPT-4.

We use the following hyper-parameters throughout all experiments:

- **temperature**: 0.0 for greedy decoding, 0.4 for self-consistent decoding;
- **max_tokens**: 1000;
- **n**: 1 for greedy decoding, 40 for self-consistent decoding;
- **frequency_penalty**: 0;
- **presence_penalty**: 0.

Any unspecified hyper-parameters are set to the default value on <https://platform.openai.com/docs/>

[api-reference/completions/create](https://platform.openai.com/docs/api-reference/completions/create) and <https://platform.openai.com/docs/api-reference/chat>.

B.1.2. Open-Source Models

We use LLaMA (7B/13B/70B) (Touvron et al., 2023) and Mistral (7B/7B-instruct) (Jiang et al., 2023) as the open-source models in our experiments. We used Nvidia A100 80GB GPUs to generate output for all open-source models. The LLaMA-70B model used 2 GPUs for each inference, while all other models used a single A100 GPU. The checkpoints and tokenizers were loaded from their respective official repositories on HuggingFace (`meta-llama` for LLaMA models and `mistralai` for Mistral models). The hyperparameters were kept the same as in the closed-source models for a fair comparison. On average, each inference took less than a second for the standard strategy, 3-4 seconds for CoT and LtM, and 5-6 seconds for FCoT and PoT for all 7B models (LLaMA-7B, Mistral-7B, and Mistral-7B-instruct). The LLaMA-13B took 1.5 times longer on average and the LLaMA-70B took 4 times longer on average. In terms of GPU hours (Nvidia A100 80GB), the LLaMA-7B, Mistral-7B, and Mistral-7B-instruct models took about 9 hours for LtM and CoT strategies, 4.5 hours for Standard, and 13 hours for PoT and FCoT strategies. In total, it took approximately 50 hours for each of the LLaMA-7B, Mistral-7B, and Mistral-7B-instruct models to run experiments for all strategies across all datasets. For LLaMA-13B it took about 75 hours and for LLaMA-70B about 200 hours. Due to the formidable computation cost of up to 425 hours, we have not finished running all baselines for open-source models yet.

B.2. Comparing Consistency Metrics

B.3. Additional Results

B.3.1. Comparing Consistency Metrics

Table B.1 compares the efficacy of three consistency metrics in terms of Brier Score averaged over all datasets and prompting strategies, with significance level. We can observe that Codex and all open-source models prefer agreement as the best or second-best (not significantly different from the best) consistency measure. GPT-3.5-turbo and GPT-4 prefer entropy and FSD, which have the same performance considering statistical significance ($p \geq 0.05$).

LM	Consistency Metrics		
	entropy	agree	FSD
Codex	.175†	.151	.159†
GPT-3.5-turbo	.205	.221†	.207
GPT-4	.116	.119†	.114
LLaMA-7B	.241†	.232	.235†
LLaMA-13B	.222†	.204	.211†
LLaMA-70B	.182†	.154	.165†
Mistral-7B	.205†	.183	.191†
Mistral-7B-instruct	.220†	.216	.215

Table B.1: Overall Brier Score (\downarrow) of three consistency metrics averaged across all datasets and prompting strategies. † indicates that the current metric is significantly worse ($p < 0.05$) than the best-performing metric (in bold).

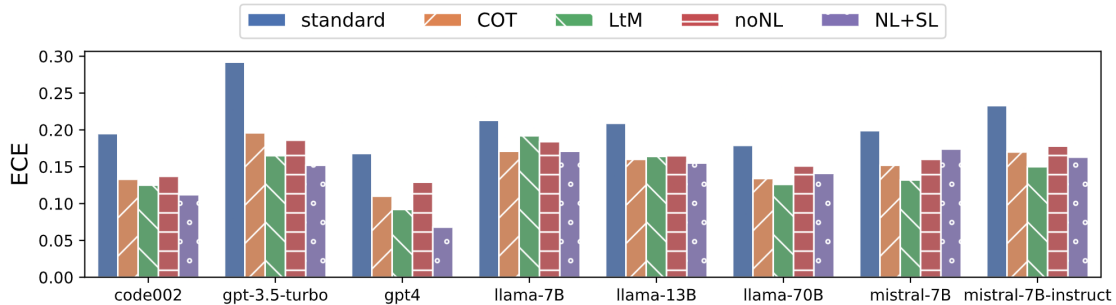


Figure B.1: ECE score (\downarrow) for each prompting strategy, averaged across all datasets and consistency metrics.

B.3.2. Calibration Results on All Datasets

Table B.2 and Table B.3 compare the Brier Score of all calibration methods for closed-source and open-source models on all 9 datasets.

B.3.3. ECE results

In addition to Brier Score, we also evaluate the calibration methods with Expected Calibration Error (ECE) (Guo et al., 2017).

Table B.4, B.5 and Figure B.1 show the ECE of all calibration methods for all models on all domains. We can observe that they exhibit similar trends as the Brier Score.

Dataset	Consistency Metrics			Baselines				
	entropy	agreement	FSD	verb _{ling}	verb _{percent}	logit	ptrue _{0-shot}	ptrue _{8-shot}
LM: Codex								
ASDiv	.099	.090	.095	.205	.190	.150	.159	.120
GSM8K	.189	.158	.177	.252	.377	.262	.248	.188
Multi	.103	.085	.089	.187	.162	.135	.106	.117
SVAMP	.126	.103	.114	.193	.173	.139	.145	.148
Sport	.071	.068	.062	.346	.075	.067	.103	.200
Date	.174	.159	.162	.210	.251	.219	.191	.197
StrategyQA	.353	.213	.256	.305	.316	.257	.271	.220
CLUTRR	.288	.369	.327	.296	.536	.506	.330	.232
SayCan	.175	.115	.152	.243	.159	.145	.135	.190
average	.175	.151	.159	.249	.249	.209	.188	.179
LM: GPT-3.5-turbo								
ASDiv	.194	.224	.194	.223	.213	n/a	n/a	n/a
GSM8K	.184	.196	.183	.260	.338	n/a	n/a	n/a
MultiArith	.044	.041	.039	.101	.065	n/a	n/a	n/a
SVAMP	.108	.115	.108	.164	.155	n/a	n/a	n/a
Sport	.089	.101	.095	.326	.151	n/a	n/a	n/a
Date	.266	.292	.280	.316	.348	n/a	n/a	n/a
StrategyQA	.393	.411	.376	.329	.342	n/a	n/a	n/a
CLUTRR	.429	.482	.465	.450	.509	n/a	n/a	n/a
SayCan	.137	.126	.126	.267	.341	n/a	n/a	n/a
average	.205	.221	.207	.271	.273	n/a	n/a	n/a
LM: GPT-4								
ASDiv	.090	.103	.090	.091	.095	n/a	n/a	n/a
GSM8K	.083	.099	.087	.132	.144	n/a	n/a	n/a
MultiArith	.013	.010	.011	.015	.013	n/a	n/a	n/a
SVAMP	.047	.050	.047	.058	.063	n/a	n/a	n/a
Sport	.033	.031	.031	.160	.100	n/a	n/a	n/a
Date	.063	.069	.061	.073	.076	n/a	n/a	n/a
StrategyQA	.230	.205	.207	.195	.220	n/a	n/a	n/a
CLUTRR	.392	.443	.416	.386	.435	n/a	n/a	n/a
SayCan	.092	.065	.079	.279	.481	n/a	n/a	n/a
average	.116	.119	.114	.154	.181	n/a	n/a	n/a

Table B.2: Brier Score (\downarrow) for closed-source LMs on all datasets, averaged across five prompting strategies. The best scores are in bold.

Domain	Consistency Metrics			Baselines logit
	entropy	agree	FSD	
LM: LLaMA-7B				
ASDiv	.164	.155	.158	.411
GSM8K	.137	.166	.148	.713
MultiArith	.232	.231	.241	.544
SVAMP	.211	.195	.211	.455
Sport	.260	.221	.232	.272
Date	.216	.267	.235	.526
StrategyQA	.390	.265	.301	.408
CLUTRR	.290	.370	.323	.633
SayCan	.267	.214	.269	.307
average	.241	.232	.235	.474
LM: LLaMA-13B				
ASDiv	.144	.135	.136	.334
GSM8K	.177	.179	.181	.591
MultiArith	.232	.198	.222	.395
SVAMP	.205	.180	.200	.365
Sport	.170	.153	.151	.194
Date	.181	.206	.185	.402
StrategyQA	.383	.241	.285	.371
CLUTRR	.298	.353	.320	.596
SayCan	.209	.190	.220	.250
average	.222	.204	.211	.389
LM: LLaMA-70B				
ASDiv	.107	.099	.101	.209
GSM8K	.201	.168	.187	.375
MultiArith	.134	.108	.113	.188
SVAMP	.145	.112	.129	.183
Sport	.053	.041	.044	.046
Date	.167	.166	.168	.262
StrategyQA	.338	.192	.239	.289
CLUTRR	.287	.347	.309	.534
SayCan	.206	.156	.191	.179
average	.182	.154	.165	.252
LM: Mistral-7B				
ASDiv	.122	.112	.113	.269
GSM8K	.197	.188	.196	.491
MultiArith	.189	.160	.171	.320
SVAMP	.176	.141	.162	.250
Sport	.116	.085	.090	.109
Date	.178	.202	.187	.373
StrategyQA	.363	.239	.276	.343
CLUTRR	.283	.334	.307	.555
SayCan	.225	.186	.217	.207
average	.205	.183	.191	.324
LM: Mistral-7B-instruct				
ASDiv	.130	.127	.124	.288
GSM8K	.193	.191	.194	.508
MultiArith	.191	.164	.180	.319
SVAMP	.166	.147	.158	.274
Sport	.207	.196	.194	.222
Date	.220	.244	.227	.497
StrategyQA	.334	.269	.284	.366
CLUTRR	.306	.397	.347	.648
SayCan	.228	.209	.229	.332
average	.220	.216	.215	.384

Table B.3: Brier Score (\downarrow) for closed-source LMs on all datasets, averaged across five prompting strategies. The best scores are **in bold**.

Domain	Consistency Metrics			Baselines				
	entropy	agreement	FSD	verb _{ling}	verb _{percent}	logit	ptrue _{0-shot}	ptrue _{8-shot}
LM: Codex								
MWP	.132	.077	.104	.237	.225	.156	.142	.108
MHQA	.188	.090	.119	.272	.214	.152	n/a	.167
Plan.	.203	.101	.159	.322	.159	.106	.117	.248
Relation.	.228	.368	.294	.214	.536	.512	.313	.175
average	.169	.117	.136	.256	.249	.189	.166	.151
LM: GPT-3.5-turbo								
MWP	.118	.121	.115	.208	.193	n/a	n/a	n/a
MHQA	.230	.246	.233	.321	.277	n/a	n/a	n/a
Plan.	.154	.119	.128	.351	.357	n/a	n/a	n/a
Relation.	.426	.505	.471	.449	.519	n/a	n/a	n/a
average	.193	.205	.195	.289	.275	n/a	n/a	n/a
LM: GPT-4								
MWP	.056	.064	.055	.053	.079	n/a	n/a	n/a
MHQA	.104	.089	.090	.139	.126	n/a	n/a	n/a
Plan.	.109	.061	.084	.351	.484	n/a	n/a	n/a
Relation.	.387	.454	.415	.381	.435	n/a	n/a	n/a
average	.114	.115	.110	.151	.179	n/a	n/a	n/a

Table B.4: ECE score (\downarrow) for closed-source LMs on four domains – Math Word Problems (MWP), Multi-hop QA (MHQA), Planning (Plan.), and Relational Inference (Relation.) – averaged across five prompting strategies. The best score is **in bold**.

Domain	Consistency Metrics			Baselines
	entropy	agreement	FSD	logit
LM: LLaMA-7B				
MWP	.138	.130	.141	.548
MHQA	.237	.189	.197	.400
Plan.	.256	.164	.259	.302
Relation.	.214	.359	.267	.641
average	.192	.179	.187	.482
LM: LLaMA-13B				
MWP	.159	.117	.151	.428
MHQA	.207	.142	.149	.319
Plan.	.205	.147	.217	.244
Relation.	.239	.334	.268	.602
average	.189	.153	.170	.391
LM: LLaMA-70B				
MWP	.153	.083	.120	.233
MHQA	.174	.082	.117	.193
Plan.	.229	.156	.207	.172
Relation.	.223	.333	.257	.539
average	.176	.118	.144	.247
LM: Mistral-7B				
MWP	.165	.111	.138	.331
MHQA	.184	.120	.135	.271
Plan.	.260	.194	.231	.198
Relation.	.207	.295	.245	.561
average	.186	.144	.159	.322
LM: Mistral-7B-instruct				
MWP	.154	.108	.127	.349
MHQA	.205	.191	.186	.355
Plan.	.203	.155	.212	.328
Relation.	.238	.402	.310	.659
average	.186	.174	.177	.383

Table B.5: ECE score (\downarrow) for all LMs on four domains – Math Word Problems (MWP), Multi-hop QA (MHQA), Planning (Plan.), and Relational Inference (Relation.) – averaged across five prompting strategies. The best score is **in bold**.

GENERATION PROMPT EXEMPLAR (ANSWER-ONLY)

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?

A: The answer is 6.

GENERATION PROMPT EXEMPLAR (CoT)

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?

A:

1. We start with 15 trees.
2. Later we have 21 trees.
3. The difference must be the number of trees they planted.
4. So, they must have planted $21 - 15 = 6$ trees.
5. The answer is 6.

GENERATION PROMPT EXEMPLAR (PoT)

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?

```
trees_begin = 15
trees_end = 21
trees_today = trees_end - trees_begin
answer = trees_today
```

GENERATION PROMPT EXEMPLAR (FAITHFUL CoT)

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?

To answer this question, write a Python program to answer the following subquestions:

1. How many trees are there in the beginning? (independent, support: ["There are 15 trees"])

```
trees_begin = 15
```

2. How many trees are there in the end? (independent, support: ["there will be 21 trees"])

```
trees_end = 21
```

3. How many trees did the grove workers plant today? (depends on 1 and 2, support: [])

```
trees_today = trees_end - trees_begin
```

4. Final Answer: How many trees did the grove workers plant today? (depends on 3, support: [])

```
answer = trees_today
```

Table C.1: An exemplar from our Generation prompt for GSM8K, with different types of output.

APPENDIX C

APPENDIX FOR CHAPTER 6

C.1. Prompts

Our prompts for each self-correction stage can be found in Table C.1 to C.6. To save space, we only show one exemplar in each prompt, but the full prompt contains 8 such exemplars in total.

DISCRIMINATION PROMPT EXEMPLAR (ANSWER-ONLY)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A: 128

Is the above answer correct? (Yes/No): No

DISCRIMINATION PROMPT EXEMPLAR (CoT)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A:

1. The width of the bathroom is 8 feet.
2. The owner wants to extend it by 2 feet on each side.
3. This means the new width will be $8 + 2 + 2 = 12$ feet.
4. The area of the bathroom is 96 sq ft.
5. The new area will be $12 * 12 = 144$ sq ft.
6. The answer is 144.

Answer: 144

Is the above answer correct? (Yes/No): No

DISCRIMINATION PROMPT EXEMPLAR (PoT)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A:

1. width_bathroom = 8
2. extension = 2
3. width_new = width_bathroom + extension * 2
4. area_bathroom = width_bathroom * width_bathroom
5. area_new = width_new * width_new
6. answer = area_new - area_bathroom

Execution result: 80

Is the above answer correct? (Yes/No): No

DISCRIMINATION PROMPT EXEMPLAR (FAITHFUL CoT)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A: # To answer this question, write a Python program to answer the following subquestions:

1. What is the width of the bathroom? (independent, support: ["the width of the bathroom is 8 feet"])

width_bathroom = 8

2. How much does the owner want to extend the bathroom on each side? (independent, support: ["extend it by 2 feet on each side"])

extension_each_side = 2

3. How much does the owner want to extend the bathroom in total? (depends on 2, support: [])

extension_total = extension_each_side * 2

4. What is the new width of the bathroom? (depends on 1 and 3, support: []) new_width_bathroom = width_bathroom + extension_total

5. What is the size of the bathroom? (independent, support: ["The size of a bathroom is 96 sq ft"])

size_bathroom = 96

6. What is the length of the bathroom? (depends on 5, support: []) length_bathroom = size_bathroom / width_bathroom

7. What is the new length of the bathroom? (depends on 6 and 3, support: [])

new_length_bathroom = length_bathroom + extension_total

8. What is the new area of the bathroom? (depends on 7 and 4, support: [])

new_area_bathroom = new_length_bathroom * new_width_bathroom

9. Final Answer: What is the new area of the bathroom? (depends on 8, support: [])

answer = new_area_bathroom

Execution result: 192

Is the above answer correct? (Yes/No): No

Table C.2: An exemplar from our Discrimination prompt for GSM8K, with different types of output.

CRITIQUING PROMPT EXEMPLAR (ANSWER-ONLY)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A: 128

Is the above answer correct? (Yes/No): No

What's wrong with it? (Please explain): The correct answer should be 140.

CRITIQUING PROMPT EXEMPLAR (CoT)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A:

1. The width of the bathroom is 8 feet.
2. The owner wants to extend it by 2 feet on each side.
3. This means the new width will be $8 + 2 + 2 = 12$ feet.
4. The area of the bathroom is 96 sq ft.
5. The new area will be $12 * 12 = 144$ sq ft.
6. The answer is 144.

Answer: 144

Is the above answer correct? (Yes/No): No

What's wrong with it? (Please explain): In Step 3, to "extend it by 2 feet on each side" means to extend the width and length by 2 feet each. So it doesn't make sense to add the extension twice to the width. Instead, we should add the extension to the width and length separately, and then multiply the new width and new length to get the area.

Table C.3: An exemplar from our Critiquing prompt for GSM8K, with different types of output (to be continued).

CRITIQUING PROMPT EXEMPLAR (PoT)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A:

```
1. width_bathroom = 8
2. extension = 2
3. width_new = width_bathroom + extension * 2
4. area_bathroom = width_bathroom * width_bathroom
5. area_new = width_new * width_new
6. answer = area_new - area_bathroom
```

Execution result: 80

Is the above answer correct? (Yes/No): No

What's wrong with it? (Please explain): In Step 3, it doesn't make sense to add the extension twice to the width. Instead, we should add the extension to the width and length separately, and then multiply the new width and new length to get the area.

CRITIQUING PROMPT EXEMPLAR (FAITHFUL CoT)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A: # To answer this question, write a Python program to answer the following subquestions:

```
# 1. What is the width of the bathroom? (independent, support: ["the width of the bathroom is 8 feet"])
width_bathroom = 8
```

```
# 2. How much does the owner want to extend the bathroom on each side? (independent, support: ["extend it by 2 feet on each side"])
extension_each_side = 2
```

```
# 3. How much does the owner want to extend the bathroom in total? (depends on 2, support: [])
extension_total = extension_each_side * 2
```

```
# 4. What is the new width of the bathroom? (depends on 1 and 3, support: []) new_width_bathroom =
width_bathroom + extension_total
```

```
# 5. What is the size of the bathroom? (independent, support: ["The size of a bathroom is 96 sq ft"])
size_bathroom = 96
```

```
# 6. What is the length of the bathroom? (depends on 5, support: []) length_bathroom = size_bathroom /
width_bathroom
```

```
# 7. What is the new length of the bathroom? (depends on 6 and 3, support: [])
new_length_bathroom = length_bathroom + extension_total
```

```
# 8. What is the new area of the bathroom? (depends on 7 and 4, support: [])
new_area_bathroom = new_length_bathroom * new_width_bathroom
```

```
# 9. Final Answer: What is the new area of the bathroom? (depends on 8, support: [])
answer = new_area_bathroom
```

Execution result: 192

Is the above answer correct? (Yes/No): No

What's wrong with it? (Please explain): In Step # 3, to "extend it by 2 feet on each side" means to extend the width and length by 2 feet each. So it doesn't make sense to compute the total extension. Instead, we should add the extension to the width and length separately, and then multiply the new width and new length to get the area.

Table C.4: An exemplar from our Critiquing prompt for GSM8K, with different types of output (continued).

EDITING PROMPT EXEMPLAR (ANSWER-ONLY)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A: 128

Is the above answer correct? (Yes/No): No

What's wrong with it? (Please explain): The correct answer should be 140.

EDITING PROMPT EXEMPLAR (CoT)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A:

1. The width of the bathroom is 8 feet.
2. The owner wants to extend it by 2 feet on each side.
3. This means the new width will be $8 + 2 + 2 = 12$ feet.
4. The area of the bathroom is 96 sq ft.
5. The new area will be $12 * 12 = 144$ sq ft.
6. The answer is 144.

Answer: 144

Is the above answer correct? (Yes/No): No

What's wrong with it? (Please explain): In Step 3, to "extend it by 2 feet on each side" means to extend the width and length by 2 feet each. So it doesn't make sense to add the extension twice to the width. Instead, we should add the extension to the width and length separately, and then multiply the new width and new length to get the area.

Let's edit the solution to fix the mistake:

1. The width of the bathroom is 8 feet.
2. The size of a bathroom is 96 sq ft, which means the length of the bathroom is $96 / 8 = 12$ feet.
3. The owner wants to extend it by 2 feet on each side.
4. This means the new width will be $8 + 2 = 10$ feet, and the new length will be $12 + 2 = 14$ feet.
5. The new area will be $10 * 14 = 140$ sq ft.
6. The answer is 140.

EDITING PROMPT EXEMPLAR (PoT)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A:

1. width_bathroom = 8
2. extension = 2
3. width_new = width_bathroom + extension * 2
4. area_bathroom = width_bathroom * width_bathroom
5. area_new = width_new * width_new
6. answer = area_new - area_bathroom

Execution result: 80

Is the above answer correct? (Yes/No): No

What's wrong with it? (Please explain): In Step 3, it doesn't make sense to add the extension twice to the width. Instead, we should add the extension to the width and length separately, and then multiply the new width and new length to get the area.

Let's edit the solution to fix the mistake:

1. width_bathroom = 8
 2. area_bathroom = 96
 3. length_bathroom = area_bathroom / width_bathroom
 4. extension = 2
 5. width_new = width_bathroom + extension
 6. length_new = length_bathroom + extension
 7. area_new = width_new * length_new
 8. answer = area_new
-

Table C.5: An exemplar from our Editing prompt for GSM8K, with different types of output (to be continued).

EDITING PROMPT EXEMPLAR (FAITHFUL CoT)

Q: The size of a bathroom is 96 sq ft. If the width of the bathroom is 8 feet and the owner wants to extend it by 2 feet on each side, what is the new area of the bathroom?

A: # To answer this question, write a Python program to answer the following subquestions:

1. What is the width of the bathroom? (independent, support: ["the width of the bathroom is 8 feet"])
width_bathroom = 8

2. How much does the owner want to extend the bathroom on each side? (independent, support: ["extend it by 2 feet on each side"])
extension_each_side = 2

3. How much does the owner want to extend the bathroom in total? (depends on 2, support: [])
extension_total = extension_each_side * 2

4. What is the new width of the bathroom? (depends on 1 and 3, support: []) new_width_bathroom = width_bathroom + extension_total

5. What is the size of the bathroom? (independent, support: ["The size of a bathroom is 96 sq ft"])
size_bathroom = 96

6. What is the length of the bathroom? (depends on 5, support: []) length_bathroom = size_bathroom / width_bathroom

7. What is the new length of the bathroom? (depends on 6 and 3, support: [])
new_length_bathroom = length_bathroom + extension_total

8. What is the new area of the bathroom? (depends on 7 and 4, support: [])
new_area_bathroom = new_length_bathroom * new_width_bathroom

9. Final Answer: What is the new area of the bathroom? (depends on 8, support: [])
answer = new_area_bathroom

Execution result: 192

Is the above answer correct? (Yes/No): No

What's wrong with it? (Please explain): In Step # 3, to "extend it by 2 feet on each side" means to extend the width and length by 2 feet each. So it doesn't make sense to compute the total extension. Instead, we should add the extension to the width and length separately, and then multiply the new width and new length to get the area.

Let's edit the solution to fix the mistake:

1. What is the width of the bathroom? (independent, support: ["the width of the bathroom is 8 feet"])
width_bathroom = 8

2. What is the size of the bathroom? (independent, support: ["The size of a bathroom is 96 sq ft"])
size_bathroom = 96

3. What is the length of the bathroom? (depends on 1 and 2, support: [])
length_bathroom = size_bathroom / width_bathroom

4. How much does the owner want to extend the bathroom on each side? (independent, support: ["extend it by 2 feet on each side"])
extension_each_side = 2

5. What is the new width of the bathroom? (depends on 1 and 4, support: [])
new_width_bathroom = width_bathroom + extension_each_side

6. What is the new length of the bathroom? (depends on 3 and 4, support: [])
new_length_bathroom = length_bathroom + extension_each_side

7. What is the new area of the bathroom? (depends on 5 and 6, support: [])
new_area_bathroom = new_length_bathroom * new_width_bathroom

8. Final Answer: What is the new area of the bathroom? (depends on 7, support: [])
answer = new_area_bathroom

Table C.6: An exemplar from our Editing prompt for GSM8K, with different types of output (continued).

APPENDIX D

APPENDIX FOR CHAPTER 7

D.1. Implementation Details

D.1.1. Models and Hyper-parameters

OpenAI Models We finetuned GPT-3 (Brown et al., 2020b) from OpenAI’s standard API³⁵ in different sizes (Davinci and Ada). Its license is MIT license. The GPT-3 models are finetuned for four epochs (default setting on the OpenAI API), and the other hyperparameters (e.g. learning rates) are the default values. with the exception of the models trained with 4k examples which were only trained for one epoch with an increased learning rate (0.2) to reduce costs.

Huggingface Models T5 (Raffel et al., 2020), BART (Lewis et al., 2020b), and OPT (Zhang et al., 2022b) are implemented with HuggingFace Transformers³⁶. The pretrained model checkpoints we use are the `t5-base` (220M parameters), `facebook/bart-base` (110M parameters) and `facebook/opt-1.3b` (1.3B parameters). Their licenses are Apache License 2.0 (T5 and BART) or other³⁷ (OPT). We use the `conditional generation` classes for T5³⁸ and BART³⁹, and use the `auto model` for `causalLM` class for OPT⁴⁰ from Huggingface to finetune the pretrained models. To remain consistent with the finetuning of OpenAI models, the T5 and BART models are finetuned with 1,000 training examples and run for 4 training epochs. The batch size is set to 8 and the learning rate is set to 2e-5 with the max sequence length being 128. The OPT model may take longer to converge, we consistently use 1,000 training examples and set batch size to 8, but the standard finetuning on CREAK, and the with-explanation finetuning on e-SNLI and ComVE run for six epochs, the learning rate of the standard finetuning on CREAK and SBIC, and the with-explanation finetuning on ComVE is set to 1e-5, the learning rate of the with-explanation finetuning on SBIC is set to 6e-5. For other

³⁵<https://beta.openai.com/docs/api-reference>

³⁶<https://github.com/huggingface/transformers>

³⁷<https://huggingface.co/facebook/opt-1.3b/blame/aa6ac1e23bb9a499be2b7400079cd2a7b8a1309a/LICENSE.md>

³⁸https://huggingface.co/docs/transformers/model_doc/t5#transformers.T5ForConditionalGeneration

³⁹https://huggingface.co/docs/transformers/model_doc/bart#transformers.BartForConditionalGeneration

⁴⁰https://huggingface.co/docs/transformers/model_doc/auto#transformers.AutoModelForCausalLM

		ComVE		CREAK		e-SNLI		SBIC	
		Standard	Explain	Standard	Explain	Standard	Explain	Standard	Explain
Accuracy (δ_{acc})	No Cue	79.2	52.4	71.6	62.6	88.0	76.4	80.0	74.6
	Sentence Length	44.8 (-34.4)	48.4 (-4.0)	53.0 (-18.6)	56.6 (-6.0)	60.4 (-27.6)	64.6 (-11.8)	53.6 (-26.4)	49.6 (-25.0)
	Present Tense	53.2 (-26.0)	54.0 (1.6)	55.2 (-16.4)	55.8 (-6.8)	67.4 (-20.6)	69.6 (-6.8)	70.6 (-9.4)	75.2 (0.6)
	Embedding Cluster	47.6 (-31.6)	48.4 (-4.0)	50.2 (-21.4)	51.8 (-10.8)	55.8 (-32.2)	58.0 (-18.4)	56.0 (-24.0)	55.0 (-19.6)
	Plural Noun	51.8 (-27.4)	53.8 (1.4)	53.0 (-18.6)	53.8 (-8.8)	52.6 (-35.4)	58.4 (-18.0)	70.8 (-9.2)	71.8 (-2.8)
	Average	49.4 (-29.9)	51.2 (-1.3)	52.9 (-18.8)	54.5 (-8.1)	59.1 (-29.0)	62.7 (-13.8)	62.8 (-17.3)	62.9 (-11.7)
	Correlation between Model’s Prediction and Spurious Feature	Sentence Length	0.870	0.778	0.847	0.590	0.644	0.531	0.676
	Present Tense	0.956	0.948	0.738	0.573	0.586	0.408	0.461	0.258
	Embedding Cluster	0.858	0.807	0.751	0.705	0.876	0.753	0.447	0.428
	Plural Noun	0.853	0.774	0.775	0.484	0.911	0.702	0.393	0.234
	Average	0.884	0.827	0.778	0.588	0.754	0.599	0.494	0.408

Table D.1: Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of GPT-3 (Ada, 2.7B), finetuned with and without explanations.

settings, the number of training epochs is set to 4 and the learning rate is set to $2e-5$. Our finetuning experiments of T5 and BART are run on a Kepler K80 GPU. The finetuning of the OPT models is run on an RTX A6000. Each finetuning takes 5 to 10 minutes depending on the task.

D.1.2. Computational Resources

All experiments performed using GPT-3 including all finetuning were performed using the OpenAI public API. We note that every finetuning experiment on each cue and dataset in this paper costs around \$10 to perform. Across all our datasets, creating a finetuned model involving 1k samples cost around \$5 when tuned without explanations and \$7 with explanations. Performing evaluation with these finetuned models then cost around a dollar when evaluating on 500 samples.

All other experiments involving heavy computational resources such as finetuning T5 and BART were performed on Google Colaboratory with GPU-accelerated notebooks available on the pro subscription.

D.2. Results for Other Models

In our main experiments in Section 7.5 and Section 7.6, we use OpenAI GPT-3 (Davinci(175B), Curie(13B), Babbage(6.7B), and Ada (2.7B), since their relatively large size may allow for the generation of higher-quality experiments, as suggested by (Wei et al., 2022a).

		ComVE		CREAK		e-SNLI		SBIC	
		Standard	Explain	Standard	Explain	Standard	Explain	Standard	Explain
Accuracy (δ_{acc})	No Cue	87.4	74.0	76.8	68.6	90.4	86.0	78.0	78.6
	Sentence Length	50.4 (-37.0)	59.0 (-15.0)	52.8 (-24.0)	59.4 (-9.2)	62.2 (-28.2)	60.6 (-25.4)	51.2 (-26.8)	52.0 (-26.6)
	Present Tense	54.2 (-33.2)	69.6 (-4.4)	55.8 (-21.0)	61.6 (-7.0)	75.0 (-15.4)	76.4 (-9.6)	73.6 (-4.4)	75.6 (-3.0)
	Embedding Cluster	50.8 (-36.6)	55.6 (-18.4)	51.8 (-25.0)	55.4 (-13.2)	63.2 (-27.2)	69.0 (-17.0)	54.8 (-23.2)	56.6 (-22.0)
	Plural Noun	52.8 (-34.6)	64.8 (-9.2)	54.4 (-22.4)	62.6 (-6.0)	59.8 (-30.6)	63.6 (-22.4)	75.8 (-2.2)	78.2 (-0.4)
	Average	52.1 (-35.4)	62.3 (-11.8)	53.7 (-23.1)	59.8 (-8.8)	65.1 (-25.4)	67.4 (-18.6)	63.9 (-14.2)	65.6 (-13.0)
	Correlation between Model’s Prediction and Spurious Feature	Sentence Length	0.821	0.524	0.894	0.659	0.633	0.582	0.753
	Present Tense	0.791	0.528	0.704	0.465	0.439	0.341	0.417	0.269
	Embedding Cluster	0.815	0.675	0.735	0.665	0.761	0.484	0.570	0.551
	Plural Noun	0.838	0.494	0.714	0.373	0.721	0.579	0.220	0.191
	Average	0.816	0.555	0.762	0.541	0.639	0.496	0.490	0.437

Table D.2: Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of GPT-3 (Babbage), finetuned with and without explanations.

We also generalize this approach to other model families including T5-base (220M), BART-base (110M), and OPT (1.3B). Table D.4 and Table D.5 show the results for these T5 and BART models respectively. Under the “no cue” condition, their performance is generally much worse than GPT-3 models. The penalty of introducing explanations in finetuning is also more striking, oftentimes resulting in an accuracy around or lower than chance (50.0). When the training set contains spurious cues, our method still generally works for both T5 and BART on three of the four datasets, as measured by $\delta_{acc}^f(M, FT)$ and $corr_f(M_f^{FT})$. However, the absolute accuracy is almost consistently lower for explanation-based finetuning than for standard finetuning, most likely due to the huge penalty under the “no cue” condition in the first place.

As an exception, on the SBIC dataset, our method does not always work well. For the T5 model, across all spurious features, explanation-based finetuning results in a similar or worse δ_{acc} (the difference is always less than 2.0 percent). It also fails to reduce the prediction-feature correlation for any spurious feature except the “embedding cluster” one, where the correlation only decreases by 0.03. For the BART model, our method does make it more robust to the “embedding cluster” and the “plural noun” cues but no other cues, as reflected by both the accuracy drop and the prediction-feature correlation. We hypothesize that this is because of the model does not rely heavily on the cues in the first place, as shown by the lower prediction-feature correlations in the case of

		ComVE		CREAK		e-SNLI		SBIC	
		Standard	Explain	Standard	Explain	Standard	Explain	Standard	Explain
Accuracy (δ_{acc})	No Cue	92.2	84.2	83.2	76.0	91.4	88.2	78.8	76.8
	Sentence Length	56.2	73.6	54.8	70.4	78.2	73.0	53.0	52.0
		(-36.0)	(-10.6)	(-28.4)	(-5.6)	(-13.2)	(-15.2)	(-25.8)	(-24.8)
	Present Tense	62.8	82.2	62.8	69.8	79.2	88.6	77.2	76.4
		(-29.4)	(-2.0)	(-20.4)	(-6.2)	(-12.2)	(-0.4)	(-1.6)	(-1.84)
	Embedding Cluster	70.0	70.6	58.2	60.6	63.4	82.6	57.8	58.4
		(-22.2)	(-13.6)	(-25.0)	(-15.4)	(-28.0)	(-5.6)	(-21.0)	(-18.4)
Plural Noun	65.0	82.2	60.0	73.0	59.0	78.4	76.2	77.0	
	(-27.2)	(-2.0)	(-23.2)	(-3.0)	(-32.4)	(-9.8)	(-2.6)	(0.2)	
Average	63.5	77.2	59.0	68.5	70.0	80.7	66.1	66.0	
	(-28.7)	(-7.1)	(-24.3)	(-7.6)	(-21.5)	(-7.6)	(-12.8)	(-10.9)	
Correlation between Model’s Prediction and Spurious Feature	Sentence Length	0.736	0.347	0.872	0.413	0.305	0.333	0.684	0.701
	Present Tense	0.756	0.244	0.589	0.402	0.364	0.075	0.244	0.231
	Embedding Cluster	0.444	0.426	0.678	0.533	0.738	0.226	0.386	0.418
	Plural Noun	0.594	0.267	0.570	0.208	0.777	0.278	0.183	0.114
Average	0.633	0.321	0.677	0.389	0.546	0.228	0.399	0.366	

Table D.3: Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of GPT-3 (Curie), finetuned with and without explanations.

standard finetuning.

We further generalize our method to OPT (1.3b) with results shown in Table D.6. Its performance under the “no cue” condition is comparable with the performance of Ada (Table D.1). Compared to standard finetuning, our method effectively mitigates the accuracy drop ($\delta_{acc}^f(M, FT)$) and the correlation between the prediction and the cue ($corr_f(M_f^{FT})$) averaged across all datasets. These results are mixed across cues however: the absolute accuracies of the with-explanation models for most tasks are lower when the “present tense” cue is introduced but are improved for all tasks in case of the “embedding cluster” cue.

Generally, compared to GPT-3, our method still works on most of the datasets for T5 and BART, but with smaller benefits. This is most likely because explanation generation is in itself a challenging task for smaller models, thus resulting in a larger penalty on accuracy in the “no cue” condition. The results of the larger OPT model lend greater credence to the validity of the assumption.

D.3. Dataset-Specific Spurious Cues

In addition to the four common spurious cues in the main text, we also construct dataset-specific spurious correlations to simulate realistic cues that can naturally appear in each dataset:

Higher Perplexity (CREAK). Using GPT-2 to measure perplexity, we filter the data into a set

		ComVE		CREAK		e-SNLI		SBIC	
		Standard	Explain	Standard	Explain	Standard	Explain	Standard	Explain
Accuracy (δ_{acc})	No Cue	76.4	49.8	55.2	41.4	86.6	55.6	69.4	65.0
	Sentence Length	53.6 (-22.8)	51.2 (1.4)	52.6 (-2.6)	45.6 (4.2)	64.0 (-22.6)	51.6 (- 4.0)	56.0 (-13.4)	53.4 (- 11.6)
	Present Tense	61.6 (-14.8)	51.2 (1.4)	50.0 (-5.2)	41.8 (0.4)	79.4 (-7.2)	42.6 (-13.0)	70.6 (1.2)	63.6 (-1.4)
	Embedding Cluster	59.4 (-17.0)	44.6 (- 5.2)	49.4 (-5.8)	38.4 (- 3.0)	69.8 (-16.8)	42.6 (- 13.0)	71.8 (2.4)	64.0 (-1.0)
	Plural Noun	73.8 (-2.6)	53.4 (3.6)	50.8 (-4.4)	40.6 (- 0.8)	59.4 (-27.2)	43.8 (- 11.8)	69.4 (0.0)	66.4 (1.4)
	Average	62.1 (-14.3)	50.1 (0.3)	50.7 (-4.5)	41.6 (0.2)	68.2 (-18.5)	45.2 (- 10.5)	67.0 (- 2.5)	61.9 (-3.2)
	Prediction- Feature Correlation	Sentence Length	0.641	0.402	0.699	0.115	0.524	0.384	0.222
	Present Tense	0.653	0.166	0.575	0.513	0.281	0.231	0.217	0.319
	Embedding Cluster	0.645	0.463	0.694	0.456	0.494	0.169	0.504	0.473
	Plural Noun	0.343	0.176	0.481	0.269	0.722	0.207	0.107	0.205
	Average	0.571	0.302	0.612	0.338	0.505	0.248	0.263	0.343

Table D.4: Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of T5-base, finetuned with and without explanations.

with above-median perplexity and a set with below-median perplexity. This feature is considered to be present if the perplexity of the sentence is higher than the median perplexity and is positively labeled.

Gender Female (e-SNLI). If the premise contains female-related pronouns (woman, women, girl, lady, etc.), we consider the “gender female” spurious cue to be present. The aforementioned words frequently appear in the e-SNLI dataset when the sentence is relevant to females.

Username Mentions (SBIC). If the social media post contains an “@” sign, meaning the author might be tagging or directly replying to other users on social media, we consider the spurious cue to be present. This feature is supposed to have no causal relationship with whether a post is offensive.

POS-tag of Swapped Word (ComVE). The ComVE dataset requires us to compare two sentences and output which sentence makes more sense, the two sentences have high lexical overlaps. We consider the part of speech (POS) of the first word which is different between the two sentences and say that the POS tag of swapped word spurious cue is present if this word is a noun.

Table D.7 shows the performance of GPT-3 (Davinci). When adding “gender female” spurious cues to

		ComVE		CREAK		e-SNLI		SBIC	
		Standard	Explain	Standard	Explain	Standard	Explain	Standard	Explain
Accuracy (δ_{acc})	No Cue	53.2	48.0	59.0	46.0	85.6	46.2	76.8	51.0
	Sentence Length	42.8 (-10.4)	43.6 (-4.4)	54.6 (-4.4)	48.4 (2.4)	54.4 (-31.2)	43.4 (-2.8)	50.8 (-26.0)	50.2 (-0.8)
	Present Tense	55.2 (2.0)	54.0 (6.0)	53.2 (-5.8)	48.0 (2.0)	58.8 (-26.8)	44.0 (-2.2)	70.8 (-6.0)	63.0 (12.0)
	Embedding Cluster	48.0 (-5.2)	47.2 (-0.8)	49.6 (-9.4)	44.0 (-2.0)	54.0 (-31.6)	40.6 (-5.6)	68.6 (-8.2)	60.0 (9.0)
	Plural Noun	54.0 (0.8)	51.2 (3.2)	53.2 (-5.8)	46.8 (0.8)	52.8 (-32.8)	48.4 (2.2)	65.2 (-11.6)	53.8 (2.8)
	Average	50.0 (-3.2)	49.0 (1.0)	52.7 (-6.4)	46.8 (0.8)	55.0 (-30.6)	44.1 (-2.1)	63.9 (-13.0)	56.8 (5.8)
	Prediction- Feature Correlation	Sentence Length	0.667	0.638	0.762	0.629	0.724	0.745	0.288
	Present Tense	0.881	0.744	0.603	0.454	0.702	0.159	0.241	0.314
	Embedding Cluster	0.817	0.792	0.801	0.700	0.854	0.301	0.555	0.395
	Plural Noun	0.823	0.230	0.607	0.491	0.884	0.439	0.287	0.210
	Average	0.797	0.601	0.693	0.569	0.791	0.411	0.343	0.406

Table D.5: Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of BART-base, finetuned with and without explanations.

the e-SNLI dataset, we find strong evidence that explanations make the model less susceptible to the spurious cue. In standard finetuning, the prediction-feature correlation is 0.684 and the accuracy is 55.8, suggesting the model relies heavily on the spurious pattern. Meanwhile, for the model finetuned with explanations, this correlation drops to 0.080, and the accuracy increases to 86.6. The results for dataset-specific cues of the ComVE and CREAK datasets are consistent with our finding that our approach is most effective when the spurious cues highly impact the model performance. On the SBIC dataset, explanation-based finetuning only decreases the prediction-feature correlation by 0.076. This could be due to the fact that the “username mention” cue is the most shallow one among all domain-specific cues, since the model only needs to detect one token (“@”), which makes it surprisingly easy for it to pick up the cue.

		ComVE		CREAK		e-SNLI		SBIC	
		Standard	Explain	Standard	Explain	Standard	Explain	Standard	Explain
Accuracy (δ_{acc})	No Cue	78.4	82.6	75.2	66.0	86.8	72.7	76.0	73.6
	Sentence Length	50.6	63.4	56.4	62.0	73.0	56.4	54.0	56.2
		(-27.8)	(-19.2)	(-18.8)	(-4.0)	(-13.8)	(-16.3)	(-22.0)	(-17.4)
	Present Tense	62.6	70.2	66.8	61.4	72.4	67.8	72.0	66.2
		(-15.8)	(-12.4)	(-8.4)	(-4.6)	(-14.4)	(-4.9)	(-4.0)	(-7.4)
	Embedding Cluster	53.4	58.6	53.6	55.2	55.4	56.0	63.6	64.8
		(-25.0)	(-24.0)	(-21.6)	(-10.8)	(-31.4)	(-16.7)	(-12.4)	(-8.8)
Plural Noun	65.6	67.2	60.0	61.6	54.8	58.2	72.6	69.8	
	(-12.8)	(-15.4)	(-15.2)	(-4.4)	(-32.0)	(-14.5)	(-3.4)	(-3.8)	
Average	58.1	64.9	59.2	60.1	63.9	59.6	65.6	64.3	
		(-20.4)	(-17.8)	(-16.0)	(-6.0)	(-22.9)	(-13.1)	(-10.5)	(-9.3)
Correlation between Model’s Prediction and Spurious Feature	Sentence Length	0.376	0.221	0.784	0.286	0.237	0.102	0.093	0.008
	Present Tense	0.686	0.282	0.499	0.437	0.419	0.331	0.224	0.187
	Embedding Cluster	0.693	0.571	0.727	0.529	0.852	0.555	0.397	0.319
	Plural Noun	0.641	0.183	0.385	0.218	0.762	0.463	0.114	0.129
	Average	0.599	0.314	0.599	0.368	0.568	0.363	0.207	0.161

Table D.6: Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of OPT (1.3b), finetuned with and without explanations.

		ComVE		CREAK		e-SNLI		SBIC	
		Standard	Explain	Standard	Explain	Standard	Explain	Standard	Explain
Accuracy (δ_{acc})	No Cue	97.0	95.6	84.2	85.0	91.6	89.2	79.0	75.0
	Domain Specific	93.6	90.4	80.5	79.0	55.8	86.6	42.6	38.3
		(-3.4)	(-5.3)	(-3.7)	(-6.0)	(-35.8)	(-2.6)	(-36.4)	(-36.7)
Prediction- Feature Correlation	Domain Specific	0.055	0.097	0.112	-0.026	0.684	0.080	0.991	0.915

Table D.7: Accuracy (\uparrow), accuracy drop (\uparrow), and prediction-feature correlation (\downarrow) on four classification tasks of GPT-3 (Davinci, 175B), finetuned with and without explanations. The skewed training sets contain domain-specific cues.

BIBLIOGRAPHY

- Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.385. URL <https://aclanthology.org/2020.acl-main.385>.
- Eldar David Abraham, Karel D’Oosterlinck, Amir Feder, Yair Ori Gat, Atticus Geiger, Christopher Potts, Roi Reichart, and Zhengxuan Wu. CEBaB: Estimating the Causal Effects of Real-World Concepts on NLP Model Behavior, 2022. URL <https://arxiv.org/abs/2205.14140>.
- Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9525–9536, 2018. URL <https://proceedings.neurips.cc/paper/2018/hash/294a8ed24b1ad22ec2e7efea049b8737-Abstract.html>.
- Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, 2022. URL <https://arxiv.org/abs/2204.01691>.
- Rohan Ajwani, Shashidhar Reddy Javaji, Frank Rudzicz, and Zining Zhu. Llm-generated black-box explanations can be adversarially helpful. *arXiv preprint arXiv:2405.06800*, 2024.
- David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7786–7795, 2018. URL <https://proceedings.neurips.cc/paper/2018/hash/3e9f0fc9b2f89e043bc6233994dfcf76-Abstract.html>.
- Afra Amini, Tiago Pimentel, Clara Meister, and Ryan Cotterell. Naturalistic Causal Probing for Morpho-Syntax, 2022. URL <https://arxiv.org/abs/2205.07043>.
- Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 39–48. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.12.

URL <https://doi.org/10.1109/CVPR.2016.12>.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-1601. URL <https://aclanthology.org/W16-1601>.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7):e0130140, 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0130140. URL <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0130140>. Publisher: Public Library of Science.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, and Katja Hansen. How to Explain Individual Classification Decisions. *Journal of Machine Learning Research*, page 29, 2010.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. Does the Whole Exceed its Parts? The Effect of AI Explanations on Complementary Team Performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pages 1–16, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 978-1-4503-8096-6. doi: 10.1145/3411764.3445717. URL <https://doi.org/10.1145/3411764.3445717>.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020. ISSN 1566-2535. doi: 10.1016/j.inffus.2019.12.012. URL <https://www.sciencedirect.com/science/article/pii/S1566253519308103>.

Jasmijn Bastings and Katja Filippova. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–155, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.blackboxnlp-1.14. URL <https://aclanthology.org/2020.blackboxnlp-1.14>.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1284. URL <https://aclanthology.org/P19-1284>.

- Samyadeep Basu, Phillip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL <https://openreview.net/forum?id=xHKVVHGDOEK>.
- Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. Identifying and controlling important neurons in neural machine translation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL <https://openreview.net/forum?id=H1z-PsR5KX>.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.
- BIG-Bench collaboration. Beyond the Imitation Game: Measuring and extrapolating the capabilities of language models, 2021. URL <https://github.com/google/BIG-bench/>.
- 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. On the Opportunities and Risks of Foundation Models. *ArXiv preprint*, abs/2108.07258, 2021. URL <https://arxiv.org/abs/2108.07258>.
- Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler,

- Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html>.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html>.
- Nitay Calderon, Eyal Ben-David, Amir Feder, and Roi Reichart. DoCoGen: Domain counterfactual generation for low resource domain adaptation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7727–7746, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.533. URL <https://aclanthology.org/2022.acl-long.533>.
- Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31, 2018a.
- Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: Natural language inference with natural language explanations. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9560–9572, 2018b. URL <https://proceedings.neurips.cc/paper/2018/hash/4c7a167bb329bd92580a99ce422d6fa6-Abstract.html>.
- R. Caruana, H. Kangarloo, J. D. Dionisio, U. Sinha, and D. Johnson. Case-based explanation of non-case-based learning methods. *Proceedings of the AMIA Symposium*, pages 212–215, 1999. ISSN 1531-605X. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2232607/>.
- Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 782–791. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.00084. URL [https://openaccess.thecvf.com/content/CVPR2021/html/Chefer_](https://openaccess.thecvf.com/content/CVPR2021/html/Chefer_Transformer_)

Interpretability_Beyond_Attention_Visualization_CVPR_2021_paper.html.

Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1223. URL <https://aclanthology.org/P16-1223>.

Howard Chen, Jacqueline He, Karthik Narasimhan, and Danqi Chen. Can rationalization improve robustness? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3792–3805. Association for Computational Linguistics, July 2022a. doi: 10.18653/v1/2022.naacl-main.278. URL <https://aclanthology.org/2022.naacl-main.278>.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code, 2021. URL <https://arxiv.org/abs/2107.03374>.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks, 2022b. URL <https://arxiv.org/abs/2211.12588>.

Yanda Chen, Ruiqi Zhong, Narutatsu Ri, Chen Zhao, He He, Jacob Steinhardt, Zhou Yu, and Kathleen McKeown. Do models explain themselves? counterfactual simulatability of natural language explanations. *arXiv preprint arXiv:2307.08678*, 2023a.

Yangyi Chen, Lifan Yuan, Ganqu Cui, Zhiyuan Liu, and Heng Ji. A Close Look into the Calibration of Pre-trained Language Models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1343–1367, Toronto, Canada, 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.75. URL <https://aclanthology.org/2023.acl-long.75>.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

- Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. A survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arXiv:2309.15402*, 2023.
- 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. Training Verifiers to Solve Math Word Problems, 2021. URL <https://arxiv.org/abs/2110.14168>.
- Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Papatankura, and Peter Clark. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.585. URL <https://aclanthology.org/2021.emnlp-main.585>.
- Fahim Dalvi, Abdul Rafae Khan, Firoj Alam, Nadir Durrani, Jia Xu, and Hassan Sajjad. Discovering latent concepts learned in BERT. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL <https://openreview.net/forum?id=POTMtpYI1xH>.
- Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3243–3255, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.262. URL <https://aclanthology.org/2020.emnlp-main.262>.
- Misha Denil, Alban Demiraj, and Nando de Freitas. Extraction of Salient Sentences from Labelled Documents. *arXiv:1412.6815 [cs]*, 2015. URL <http://arxiv.org/abs/1412.6815>. arXiv: 1412.6815.
- Daniel Deutsch, Shyam Upadhyay, and Dan Roth. A general-purpose algorithm for constrained sequential inference. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 482–492, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/K19-1045. URL <https://aclanthology.org/K19-1045>.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- Finale Doshi-Velez and Been Kim. Towards A Rigorous Science of Interpretable Machine Learning. *ArXiv preprint*, abs/1702.08608, 2017. URL <https://arxiv.org/abs/1702.08608>.
- Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral

- explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175, 2021. doi: 10.1162/tacl_a_00359. URL <https://aclanthology.org/2021.tacl-1.10>.
- Amir Feder, Nadav Oved, Uri Shalit, and Roi Reichart. CausaLM: Causal model explanation through counterfactual language models. *Computational Linguistics*, 47(2):333–386, 2021. doi: 10.1162/coli_a_00404. URL <https://aclanthology.org/2021.cl-2.13>.
- Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1407. URL <https://aclanthology.org/D18-1407>.
- Hao Fu, Yao; Peng and Tushar Khot. How does gpt obtain its ability? tracing emergent abilities of language models to their sources. *Yao Fu’s Notion*, Dec 2022. URL <https://yaofu.notion.site/How-does-GPT-Obtain-its-Ability-Tracing-Emergent-Abilities-of-Language-Models-to-their-Sources-b9a57ac0fc7>
- Yarin Gal. Uncertainty in deep learning. 2016. URL <https://api.semanticscholar.org/CorpusID:86522127>.
- Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059. JMLR.org, 2016. URL <http://proceedings.mlr.press/v48/gal16.html>.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided Language Models, 2022. URL <https://arxiv.org/abs/2211.10435>.
- Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koepl, Preslav Nakov, and Iryna Gurevych. A Survey of Language Model Confidence Estimation and Calibration, 2023. URL <https://arxiv.org/abs/2311.08298>.
- Mor Geva, Yoav Goldberg, and Jonathan Berant. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1107. URL <https://aclanthology.org/D19-1107>.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. *Did Aristotle Use a Laptop?* A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021. ISSN 2307-387X.

- doi: 10.1162/tacl_a_00370. URL https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00370/100680/Did-Aristotle-Use-a-Laptop-A-Question-Answering.
- Amirata Ghorbani, Abubakar Abid, and James Y. Zou. Interpretation of neural networks is fragile. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3681–3688. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33013681. URL <https://doi.org/10.1609/aaai.v33i01.33013681>.
- Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017. URL <http://proceedings.mlr.press/v70/guo17a.html>.
- Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. Neural module networks for reasoning over text. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL <https://openreview.net/forum?id=SygWvAVFPr>.
- Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2017. URL <https://aclanthology.org/N18-2017>.
- Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46:389–422, 2002.
- Joseph Y. Halpern and Judea Pearl. Causes and Explanations: A Structural-Model Approach. Part I: Causes. *The British Journal for the Philosophy of Science*, 56(4):843–887, 2005. ISSN 0007-0882. doi: 10.1093/bjps/axi147. URL <https://www.journals.uchicago.edu/doi/10.1093/bjps/axi147>. Publisher: The University of Chicago Press.
- Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.492. URL <https://aclanthology.org/2020.acl-main.492>.
- Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Self-attention attribution: Interpreting information interactions inside transformer. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021*,

- The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12963–12971. AAAI Press, 2021. URL <https://ojs.aaai.org/index.php/AAAI/article/view/17533>.
- L. A. Harrington, M. D. Morley, A. Šcedrov, and S. G. Simpson. *Harvey Friedman’s Research on the Foundations of Mathematics*. Elsevier, 1985. ISBN 978-0-08-096040-1. Google-Books-ID: 2pIPRR4LDxIC.
- Peter Hase and Mohit Bansal. When can models learn from explanations? a formal framework for understanding the roles of explanation data. In *Proceedings of the First Workshop on Learning with Natural Language Supervision*, pages 29–39, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.lnls-1.4. URL <https://aclanthology.org/2022.lnls-1.4>.
- Peter Hase, Shiyue Zhang, Harry Xie, and Mohit Bansal. Leakage-adjusted simulatability: Can models generate non-trivial explanations of their behavior in natural language? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4351–4367, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.390. URL <https://aclanthology.org/2020.findings-emnlp.390>.
- Bernease Herman. The Promise and Peril of Human Evaluation for Model Interpretability. *ArXiv preprint*, abs/1711.07414, 2017. URL <https://arxiv.org/abs/1711.07414>.
- Avery Hiebert, Cole Peterson, Alona Fyshe, and Nishant Mehta. Interpreting word-level hidden state behaviour of character-level LSTM language models. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 258–266, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5428. URL <https://aclanthology.org/W18-5428>.
- Ruixin Hong, Hongming Zhang, Xintong Yu, and Changshui Zhang. METGEN: A module-based entailment tree generation framework for answer explanation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1887–1905, Seattle, United States, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.145. URL <https://aclanthology.org/2022.findings-naacl.145>.
- Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 804–813. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.93. URL <https://doi.org/10.1109/ICCV.2017.93>.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.
- Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable NLP systems: How should we

- define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.386. URL <https://aclanthology.org/2020.acl-main.386>.
- Alon Jacovi and Yoav Goldberg. Aligning faithful interpretations with their social attribution. *Transactions of the Association for Computational Linguistics*, 9:294–310, 2021. doi: 10.1162/tacl_a_00367. URL <https://aclanthology.org/2021.tacl-1.18>.
- Alon Jacovi, Swabha Swayamdipta, Shauli Ravfogel, Yanai Elazar, Yejin Choi, and Yoav Goldberg. Contrastive explanations for model interpretability. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1597–1611, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.120. URL <https://aclanthology.org/2021.emnlp-main.120>.
- Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1357. URL <https://aclanthology.org/N19-1357>.
- Joseph D. Janizek, Pascal Sturmfels, and Su-In Lee. Explaining explanations: Axiomatic feature interactions for deep networks. *J. Mach. Learn. Res.*, 22:104:1–104:54, 2021. URL <http://jmlr.org/papers/v22/20-1223.html>.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth ee Lacroix, and William El Sayed. Mistral 7b, 2023.
- Chengyue Jiang, Yinggong Zhao, Shanbo Chu, Libin Shen, and Kewei Tu. Cold-start and interpretability: Turning regular expressions into trainable recurrent neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3193–3207, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.258. URL <https://aclanthology.org/2020.emnlp-main.258>.
- Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021. doi: 10.1162/tacl_a_00407. URL <https://aclanthology.org/2021.tacl-1.57>.
- Brihi Joshi, Ziyi Liu, Sahana Ramnath, Aaron Chan, Zhewei Tong, Shaoliang Nie, Qifan Wang, Yejin Choi, and Xiang Ren. Are machine rationales (not) useful to humans? measuring and improving human utility of free-text rationales. *arXiv preprint arXiv:2305.07095*, 2023.
- Jaehun Jung, Lianhui Qin, Sean Welleck, Faeze Brahman, Chandra Bhagavatula, Ronan Le Bras,

- and Yejin Choi. Maieutic prompting: Logically consistent reasoning with recursive explanations. *arXiv preprint arXiv:2205.11822*, 2022a.
- Jaehun Jung, Lianhui Qin, Sean Welleck, Faeze Brahman, Chandra Bhagavatula, Ronan Le Bras, and Yejin Choi. Maieutic prompting: Logically consistent reasoning with recursive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1266–1279, Abu Dhabi, United Arab Emirates, 2022b. Association for Computational Linguistics. URL <https://aclanthology.org/2022.emnlp-main.82>.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022.
- Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In Kristina Toutanova and Hua Wu, editors, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1062. URL <https://aclanthology.org/P14-1062>.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. End-to-end bias mitigation by modelling biases in corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8706–8716, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.769. URL <https://aclanthology.org/2020.acl-main.769>.
- Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and Understanding Recurrent Networks. *ArXiv preprint*, abs/1506.02078, 2015. URL <https://arxiv.org/abs/1506.02078>.
- Divyansh Kaushik and Zachary C. Lipton. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015. Association for Computational Linguistics, October–November 2018a. doi: 10.18653/v1/D18-1546. URL <https://aclanthology.org/D18-1546>.
- Divyansh Kaushik and Zachary C. Lipton. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, Brussels, Belgium, 2018b. Association for Computational Linguistics. doi: 10.18653/v1/D18-1546. URL <https://aclanthology.org/>

D18-1546.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. Learning the difference that makes a difference with counterfactually-augmented data. *arXiv preprint arXiv:1909.12434*, 2019.

Divyansh Kaushik, Eduard H. Hovy, and Zachary Chase Lipton. Learning the difference that makes A difference with counterfactually-augmented data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL <https://openreview.net/forum?id=SkLgs0NFvr>.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (Un)reliability of Saliency Methods. In Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, editors, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700, pages 267–280. Springer International Publishing, Cham, 2019. ISBN 978-3-030-28953-9 978-3-030-28954-6. doi: 10.1007/978-3-030-28954-6_14. URL http://link.springer.com/10.1007/978-3-030-28954-6_14. Series Title: Lecture Notes in Computer Science.

Svetlana Kiritchenko and Saif Mohammad. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 43–53, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-2005. URL <https://aclanthology.org/S18-2005>.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 2017. URL <http://proceedings.mlr.press/v70/koh17a.html>.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners, 2022. URL <https://arxiv.org/abs/2205.11916>.

Sawan Kumar and Partha Talukdar. NILE : Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8730–8742, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.771. URL <https://aclanthology.org/2020.acl-main.771>.

Johannes Kunkel, Tim Donkers, Lisa Michael, Catalin-Mihai Barbu, and Jürgen Ziegler. Let me explain: Impact of personal and impersonal explanations on trust in recommender systems. In Stephen A. Brewster, Geraldine Fitzpatrick, Anna L. Cox, and Vassilis Kostakos, editors, *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019*, page 487. ACM, 2019. doi: 10.1145/3290605.3300717. URL <https://doi.org/10.1145/3290605.3300717>.

Himabindu Lakkaraju and Osbert Bastani. "How do I fool you?": Manipulating User Trust

- via Misleading Black Box Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 79–85, New York NY USA, 2020. ACM. ISBN 978-1-4503-7110-0. doi: 10.1145/3375627.3375833.
- Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6402–6413, 2017. URL <https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html>.
- Andrew Lampinen, Ishita Dasgupta, Stephanie Chan, Kory Mathewson, Mh Tessler, Antonia Creswell, James McClelland, Jane Wang, and Felix Hill. Can language models learn from explanations in context? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 537–563, Abu Dhabi, United Arab Emirates, 2022a. Association for Computational Linguistics. URL <https://aclanthology.org/2022.findings-emnlp.38>.
- Andrew K. Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L. McClelland, Jane X. Wang, and Felix Hill. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329*, 2022b.
- Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.
- Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7167–7177, 2018. URL <https://proceedings.neurips.cc/paper/2018/hash/abdeb6f575ac5c6676b747bca8d09cc2-Abstract.html>.
- Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas, 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1011. URL <https://aclanthology.org/D16-1011>.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL <https://aclanthology.org/2020.acl-main.703>.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer

- Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880. Association for Computational Linguistics, July 2020b. doi: 10.18653/v1/2020.acl-main.703. URL <https://aclanthology.org/2020.acl-main.703>.
- Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California, 2016a. Association for Computational Linguistics. doi: 10.18653/v1/N16-1082. URL <https://aclanthology.org/N16-1082>.
- Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding Neural Networks through Representation Erasure. *ArXiv preprint*, abs/1612.08220, 2016b. URL <https://arxiv.org/abs/1612.08220>.
- Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. On the Advance of Making Language Models Better Reasoners, 2022. URL <https://arxiv.org/abs/2206.02336>.
- Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL <https://openreview.net/forum?id=8s8K2UZGTZ>.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1015. URL <https://aclanthology.org/P17-1015>.
- Zachary C. Lipton. The Mythos of Model Interpretability. *ArXiv preprint*, abs/1606.03490, 2016. URL <https://arxiv.org/abs/1606.03490>.
- Haochen Liu, Joseph Thekinen, Sinem Mollaoglu, Da Tang, Ji Yang, Youlong Cheng, Hui Liu, and Jiliang Tang. Toward annotator group bias in crowdsourcing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1797–1806, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.126. URL <https://aclanthology.org/2022.acl-long.126>.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.
- Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. Gender bias in neural natural language processing. In *Logic, Language, and Security*, pages 189–202. Springer, 2020.

- Kaiji Lu, Zifan Wang, Piotr Mardziel, and Anupam Datta. Influence patterns for explaining information flow in BERT. In Marc’Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 4461–4474, 2021. URL <https://proceedings.neurips.cc/paper/2021/hash/239f914f30ea3c948fce2ea07a9efb33-Abstract.html>.
- Josh Magnus Ludan, Yixuan Meng, Tai Nguyen, Saurabh Shah, Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. Explanation-based finetuning makes models more robust to spurious cues. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4420–4441, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.242. URL <https://aclanthology.org/2023.acl-long.242>.
- Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774, 2017. URL <https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html>.
- Qing Lyu, Hongming Zhang, Elior Sulem, and Dan Roth. Zero-shot event extraction via transfer learning: Challenges and insights. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 322–332, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.42. URL <https://aclanthology.org/2021.acl-short.42>.
- Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. Towards Faithful Model Explanation in NLP: A Survey, 2022a. URL <https://arxiv.org/abs/2209.11326>.
- Qing Lyu, Zheng Hua, Daoxin Li, Li Zhang, Marianna Apidianaki, and Chris Callison-Burch. Is “My Favorite New Movie” My Favorite Movie? Probing the Understanding of Recursive Noun Phrases. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5286–5302, Seattle, United States, 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.388. URL <https://aclanthology.org/2022.naacl-main.388>.
- Qing Lyu, Marianna Apidianaki, and Chris Callison-burch. Representation of lexical stylistic features in language models’ embedding space. In Alexis Palmer and Jose Camacho-collados, editors, *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 370–387, Toronto, Canada, 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.starsem-1.32. URL <https://aclanthology.org/2023.starsem-1.32>.
- Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful Chain-of-Thought Reasoning, 2023b. URL <https://arxiv.org/>

abs/2301.13379.

- Qing Lyu, Kumar Shridhar, Chaitanya Malaviya, Li Zhang, Yanai Elazar, Niket Tandon, Marianna Apidianaki, Mrinmaya Sachan, and Chris Callison-Burch. Calibrating large language models with sample consistency, 2024.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *ArXiv preprint*, abs/2303.17651, 2023. URL <https://arxiv.org/abs/2303.17651>.
- Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL <https://aclanthology.org/2023.emnlp-main.557>.
- Ana Marasovic, Iz Beltagy, Doug Downey, and Matthew Peters. Few-shot self-rationalization with natural language prompts. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 410–424. Association for Computational Linguistics, July 2022a. doi: 10.18653/v1/2022.findings-naacl.31. URL <https://aclanthology.org/2022.findings-naacl.31>.
- Ana Marasovic, Iz Beltagy, Doug Downey, and Matthew Peters. Few-shot self-rationalization with natural language prompts. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 410–424, Seattle, United States, 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.31. URL <https://aclanthology.org/2022.findings-naacl.31>.
- Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448. Association for Computational Linguistics, July 2019a. doi: 10.18653/v1/P19-1334. URL <https://aclanthology.org/P19-1334>.
- Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy, 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1334. URL <https://aclanthology.org/P19-1334>.
- Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.92. URL <https://aclanthology.org/2020.acl-main.92>.
- Sabrina J. Mielke, Arthur Szlam, Emily Dinan, and Y-Lan Boureau. Reducing conversational agents’ overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics*, 10:857–872, 2022. doi: 10.1162/tacl_a_00494. URL <https://aclanthology.org/2022>.

tacl-1.50.

Tim Miller. Explanation in Artificial Intelligence: Insights from the Social Sciences. *ArXiv preprint*, abs/1706.07269, 2017. URL <https://arxiv.org/abs/1706.07269>.

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 65:211–222, 2017. ISSN 0031-3203. doi: 10.1016/j.patcog.2016.11.008. URL <https://www.sciencedirect.com/science/article/pii/S0031320316303582>.

W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019. doi: 10.1073/pnas.1900654116. URL <https://www.pnas.org/doi/10.1073/pnas.1900654116>. Publisher: Proceedings of the National Academy of Sciences.

Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. WT5?! Training Text-to-Text Models to Explain their Predictions. *ArXiv preprint*, abs/2004.14546, 2020. URL <https://arxiv.org/abs/2004.14546>.

Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3806–3815. PMLR, 2018. URL <http://proceedings.mlr.press/v80/nie18a.html>.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.441. URL <https://aclanthology.org/2020.acl-main.441>.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show Your Work: Scratchpads for Intermediate Computation with Language Models, 2021a. URL <https://arxiv.org/abs/2112.00114>.

Maxwell I. Nye, Michael Henry Tessler, Joshua B. Tenenbaum, and Brenden M. Lake. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. In Marc’Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 25192–25204, 2021b. URL <https://proceedings.neurips.cc/paper/2021/hash/d3e2e8f631bd9336ed25b8162aef8782-Abstract.html>.

Yasumasa Onoe, Michael JQ Zhang, Eunsol Choi, and Greg Durrett. Creak: A dataset for

- commonsense reasoning over entity knowledge. *ArXiv preprint*, abs/2109.01653, 2021. URL <https://arxiv.org/abs/2109.01653>.
- OpenAI. GPT-4 Technical Report. *ArXiv preprint*, abs/2303.08774, 2023. URL <https://arxiv.org/abs/2303.08774>.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL <https://aclanthology.org/2021.naacl-main.168>.
- Nina Poerner, Benjamin Roth, and Hinrich Schütze. Interpretable textual neuron representations for NLP. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 325–327, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5437. URL <https://aclanthology.org/W18-5437>.
- Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-2023. URL <https://aclanthology.org/S18-2023>.
- Gwenyth Portillo Wightman, Alexandra Delucia, and Mark Dredze. Strength in numbers: Estimating confidence of large language models by prompt agreement. In Anaelia Ovalle, Kai-Wei Chang, Ninareh Mehrabi, Yada Pruksachatkun, Aram Galystan, Jwala Dhamala, Apurv Verma, Trista Cao, Anoop Kumar, and Rahul Gupta, editors, *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 326–362, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.trustnlp-1.28. URL <https://aclanthology.org/2023.trustnlp-1.28>.
- Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C. Lipton. Learning to deceive with attention-based explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4782–4793, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.432. URL <https://aclanthology.org/2020.acl-main.432>.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. page 24, 2019.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>.

- Dheeraj Rajagopal, Vidhisha Balachandran, Eduard H Hovy, and Yulia Tsvetkov. SELFEXPLAIN: A self-explaining architecture for neural text classifiers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 836–850, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.64. URL <https://aclanthology.org/2021.emnlp-main.64>.
- Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1487. URL <https://aclanthology.org/P19-1487>.
- Franco Rajič, Ivan Stresec, Axel Marmet, and Tim Poštuvan. Using focal loss to fight shallow heuristics: An empirical analysis of modulated cross-entropy in natural language inference. *ArXiv preprint*, abs/2211.13331, 2022. URL <https://arxiv.org/abs/2211.13331>.
- A Rakhlin. Convolutional neural networks for sentence classification. *GitHub*, 6:25, 2016.
- Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.647. URL <https://aclanthology.org/2020.acl-main.647>.
- Shauli Ravfogel, Grusha Prasad, Tal Linzen, and Yoav Goldberg. Counterfactual interventions reveal the causal effect of relative clause representations on agreement prediction. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 194–209, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.conll-1.15. URL <https://aclanthology.org/2021.conll-1.15>.
- Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL <https://aclanthology.org/D19-1410>.
- Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM, 2016. doi: 10.1145/2939672.2939778. URL <https://doi.org/10.1145/2939672.2939778>.
- Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-*

- Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1527–1535. AAAI Press, 2018. URL <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982>.
- Neal J. Roese and James M. Olson. Counterfactual thinking: A critical overview. In *What might have been: The social psychology of counterfactual thinking*, pages 1–55. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, US, 1995. ISBN 978-0-8058-1613-6 978-0-8058-1614-3.
- Alexis Ross, Matthew E. Peters, and Ana Marasović. Does self-rationalization improve robustness to spurious correlations? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, November 2022. URL <https://aclanthology.org/2022.emnlp-main.501.pdf>.
- Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1202. URL <https://aclanthology.org/D15-1202>.
- Hassan Sajjad, Nadir Durrani, and Fahim Dalvi. Neuron-level interpretation of deep NLP models: A survey. *Transactions of the Association for Computational Linguistics*, 10:1285–1303, 2022. doi: 10.1162/tacl_a_00519. URL <https://aclanthology.org/2022.tacl-1.74>.
- Victor Sanh, Thomas Wolf, Yonatan Belinkov, and Alexander M. Rush. Learning from others’ mistakes: Avoiding dataset biases without modeling them. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL <https://openreview.net/forum?id=Hf3qXoiNkR>.
- Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A. Smith, and Yejin Choi. Social bias frames: Reasoning about social and power implications of language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5477–5490, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.486. URL <https://aclanthology.org/2020.acl-main.486>.
- Gabriele Sarti, Nils Feldhus, Ludwig Sickert, and Oskar van der Wal. Inseq: An interpretability toolkit for sequence generation models. *ArXiv preprint*, abs/2302.13942, 2023. URL <https://arxiv.org/abs/2302.13942>.
- William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.
- Roy Schwartz, Sam Thomson, and Noah A. Smith. Bridging CNNs, RNNs, and weighted finite-state machines. In *Proceedings of the 56th Annual Meeting of the Association for Computational*

- Linguistics (Volume 1: Long Papers)*, pages 295–305, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1028. URL <https://aclanthology.org/P18-1028>.
- Kumar Shridhar, Harsh Jhamtani, Hao Fang, Benjamin Van Durme, Jason Eisner, and Patrick Xia. Screws: A modular framework for reasoning with revisions. *ArXiv preprint*, abs/2309.13075, 2023. URL <https://arxiv.org/abs/2309.13075>.
- Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 2017. URL <http://proceedings.mlr.press/v70/shrikumar17a.html>.
- Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *In Workshop at International Conference on Learning Representations*, 2014.
- Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1458. URL <https://aclanthology.org/D19-1458>.
- Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 978-1-4503-7110-0. URL <https://doi.org/10.1145/3375627.3375830>.
- Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *ArXiv preprint*, abs/1706.03825, 2017. URL <https://arxiv.org/abs/1706.03825>.
- J Springenberg, Alexey Dosovitskiy, Thomas Brox, and M Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.
- Joe Stacey, Pasquale Minervini, Haim Dubossarsky, Sebastian Riedel, and Tim Rocktäschel. There is strength in numbers: Avoiding the hypothesis-only bias in natural language inference via ensemble adversarial training. 2020.
- Joe Stacey, Yonatan Belinkov, and Marek Rei. Supervising model attention with human explanations for robust natural language inference. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelfth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 11349–11357. AAAI Press, 2022. URL

<https://ojs.aaai.org/index.php/AAAI/article/view/21386>.

Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M. Rush. LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676, 2018. ISSN 1941-0506. doi: 10.1109/TVCG.2017.2744158. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

Sanjay Subramanian, Ben Bogin, Nitish Gupta, Tomer Wolfson, Sameer Singh, Jonathan Berant, and Matt Gardner. Obtaining faithful interpretations from compositional neural networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5594–5608, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.495. URL <https://aclanthology.org/2020.acl-main.495>.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017. URL <http://proceedings.mlr.press/v70/sundararajan17a.html>.

Oyvind Taffjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.317. URL <https://aclanthology.org/2021.findings-acl.317>.

Derek Tam, Anisha Mascarenhas, Shiyue Zhang, Sarah Kwan, Mohit Bansal, and Colin Raffel. Evaluating the factual consistency of large language models through summarization. *ArXiv preprint*, abs/2211.08412, 2022. URL <https://arxiv.org/abs/2211.08412>.

David Thomas. Pragmatic programmer, the: From journeyman to master andrew hunt. page 95, 1999.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*,

- abs/2307.09288, 2023. URL <https://arxiv.org/abs/2307.09288>.
- Michael Tsang, Sirisha Rambhatla, and Yan Liu. How does this interaction affect me? interpretable attribution for feature interactions. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL <https://proceedings.neurips.cc/paper/2020/hash/443dec3062d0286986e21dc0631734c9-Abstract.html>.
- Mycal Tucker, Peng Qian, and Roger Levy. What if this modified that? syntactic interventions with counterfactual embeddings. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 862–875, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.76. URL <https://aclanthology.org/2021.findings-acl.76>.
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don’t always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=bzs4uPLXvi>.
- Martin Tutek and Jan Snajder. Staying true to your word: (how) can attention become explanation? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 131–142, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.repl4nlp-1.17. URL <https://aclanthology.org/2020.repl4nlp-1.17>.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>.
- Jesse Vig. Visualizing Attention in Transformer-Based Language Representation Models. *ArXiv preprint*, abs/1904.02679, 2019. URL <https://arxiv.org/abs/1904.02679>.
- Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. Investigating gender bias in language models using causal mediation analysis. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL <https://proceedings.neurips.cc/paper/2020/hash/92650b2e92217715fe312e6fa7b90d82-Abstract.html>.
- Eric Wallace, Shi Feng, and Jordan Boyd-Graber. Interpreting neural networks with nearest neighbors. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 136–144, Brussels, Belgium, 2018. Association for Computational

- Linguistics. doi: 10.18653/v1/W18-5416. URL <https://aclanthology.org/W18-5416>.
- Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1221. URL <https://aclanthology.org/D19-1221>.
- Eric Wallace, Matt Gardner, and Sameer Singh. Interpreting predictions of NLP models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pages 20–23, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-tutorials.3. URL <https://aclanthology.org/2020.emnlp-tutorials.3>.
- Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. *ArXiv preprint*, abs/2212.10001, 2022a. URL <https://arxiv.org/abs/2212.10001>.
- Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. Does it make sense? and why? a pilot study for sense making and explanation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4020–4026, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1393. URL <https://aclanthology.org/P19-1393>.
- Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, et al. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *ArXiv preprint*, abs/2310.07521, 2023. URL <https://arxiv.org/abs/2310.07521>.
- Junlin Wang, Jens Tuyls, Eric Wallace, and Sameer Singh. Gradient-based analysis of NLP models is manipulable. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 247–258, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.24. URL <https://aclanthology.org/2020.findings-emnlp.24>.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models, 2022b. URL <https://arxiv.org/abs/2203.11171>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022a.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain of Thought Prompting Elicits Reasoning in Large Language Models, 2022b. URL <https://arxiv.org/abs/2201.11903>.

- Sarah Wiegrefe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1002. URL <https://aclanthology.org/D19-1002>.
- Sarah Wiegrefe, Ana Marasović, and Noah A. Smith. Measuring association between labels and free-text rationales. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10266–10284, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL <https://aclanthology.org/2021.emnlp-main.804>.
- Sarah Wiegrefe, Jack Hessel, Swabha Swayamdipta, Mark Riedl, and Yejin Choi. Reframing human-AI collaboration for generating free-text explanations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 632–658, Seattle, United States, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.47. URL <https://aclanthology.org/2022.naacl-main.47>.
- Christopher Winship and Stephen L. Morgan. The Estimation of Causal Effects from Observational Data. *Annual Review of Sociology*, 25(1):659–706, 1999. doi: 10.1146/annurev.soc.25.1.659. URL <https://doi.org/10.1146/annurev.soc.25.1.659>. _eprint: <https://doi.org/10.1146/annurev.soc.25.1.659>.
- Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6707–6723, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.523. URL <https://aclanthology.org/2021.acl-long.523>.
- Yuxiang Wu, Matt Gardner, Pontus Stenetorp, and Pradeep Dasigi. Generating data to mitigate spurious correlations in natural language inference datasets. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2660–2676, Dublin, Ireland, 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.190. URL <https://aclanthology.org/2022.acl-long.190>.
- Zhengxuan Wu, Atticus Geiger, Joshua Rozner, Elisa Kreiss, Hanson Lu, Thomas Icard, Christopher Potts, and Noah Goodman. Causal distillation for language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4295, Seattle, United States, 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.318. URL <https://aclanthology.org/2022.naacl-main.318>.
- Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *ArXiv preprint, abs/2306.13063*, 2023. URL <https://arxiv.org/abs/2306.13063>.

- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.
- Xi Ye and Greg Durrett. The Unreliability of Explanations in Few-shot Prompting for Textual Reasoning, 2022. URL <https://arxiv.org/abs/2205.03401>.
- Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: Satisfiability-aided language models using declarative prompting. *Advances in Neural Information Processing Systems*, 36, 2024.
- Kayo Yin and Graham Neubig. Interpreting language models with contrastive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 184–198, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.emnlp-main.14>.
- KiYoon Yoo, Jangho Kim, Jiho Jang, and Nojun Kwak. Detection of Word Adversarial Examples in Text Classification: Benchmark and Baseline via Robust Density Estimation, 2022. URL <https://arxiv.org/abs/2203.01677>.
- Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8689, pages 818–833. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10589-5 978-3-319-10590-1. doi: 10.1007/978-3-319-10590-1_53. URL http://link.springer.com/10.1007/978-3-319-10590-1_53. Series Title: Lecture Notes in Computer Science.
- Hanlin Zhang, Ziyang Li, Jiani Huang, Mayur Naik, and Eric Xing. Improved logical reasoning of language models via differentiable symbolic programming. In *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*, 2022a. URL <https://openreview.net/forum?id=8lNy3QCaxHX>.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022b.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models, 2022. URL <https://arxiv.org/abs/2205.10625>.
- Yilun Zhou and Julie Shah. The solvability of interpretability evaluation metrics. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2399–2415, Dubrovnik, Croatia, 2023. Association for Computational Linguistics. URL <https://aclanthology.org/2023.findings-eacl.182>.
- Chiwei Zhu, Benfeng Xu, Quan Wang, Yongdong Zhang, and Zhendong Mao. On the calibration of

- large language models and alignment. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9778–9795, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.654. URL <https://aclanthology.org/2023.findings-emnlp.654>.
- Zining Zhu, Hanjie Chen, Xi Ye, Qing Lyu, Chenhao Tan, Ana Marasovic, and Sarah Wiegrefe. Explanation in the era of large language models. In Rui Zhang, Nathan Schneider, and Snigdha Chaturvedi, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 5: Tutorial Abstracts)*, pages 19–25, Mexico City, Mexico, June 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.naacl-tutorials.3>.
- Ran Zmigrod, Sabrina J. Mielke, Hanna Wallach, and Ryan Cotterell. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1651–1661, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1161. URL <https://aclanthology.org/P19-1161>.