The Dissertation Committee for Xi Ye
certifies that this is the approved version of the following dissertation:

Steering Textual Reasoning with Explanations

Committee:

Greg Durret, Supervisor

Raymond Mooney

Eunsol Choi

Jonathan Berant

Steering Textual Reasoning with Explanations

by
Xi Ye

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin
May 2024

Acknowledgments

I have been fortunate to pursue my PhD studies abroad at UT, and I have been incredibly lucky throughout my doctoral journey.

First and foremost, I would like to thank my advisor, Greg Durrett. I wasn't initially admitted as his student but decided to take a turn to NLP after I joined UT. I am deeply grateful that although I didn't have much background in NLP, he let me join his group. Ever since then, it has been an absolutely wonderful journey. Greg is so passionate, knowledgeable, and creative as a researcher, as well as thoughtful and helpful as an advisor. He can patiently answer my questions like "What is language modeling" during our meetings, and he is willing to look at the lengthy, messy output examples and examine my figures at the pixel level. He allows me almost complete flexibility in choosing the topics I am interested in, and in taking internships and external collaborations. Even more incredibly, Greg is always encouraging and supportive whenever I panic about projects or anything. Greg is my role model not only as a leading researcher in NLP but also as a supportive mentor. I could never express how fortunate I am to be his student.

I am also grateful to Qixing Huang, without whose support I would not have been able to pursue my Ph.D. studies at UT. I first met with Qixing during his visit to Tsinghua University. I am thankful that he hosted me for a summer visit at UT, even though I was not particularly strong as a CV student. I found myself enjoying UT and the city even during summer (probably the worst time to stay in Austin). The visiting experience definitely played an important part in my admission to UT a year later. I also appreciate Qixing's advice and caring throughout my years at UT.

I would like to extend my gratitude to all the mentors who generously shared their time and advice. Eunsol Choi has provided enormous help, offering insightful feedback on my projects and papers, collaboration opportunities, and kind words of encouragement (as well as recommendations for Korean restaurants which have become my go-to places for celebrating paper acceptances). Eunsol has also been a fantastic example of how to be a great advisor. I am sincerely grateful to Isil Dillig for her collaboration and mentorship on many interesting interdisciplinary projects. From Isil, I have also learned a great deal about formalizing problems and writing papers. I would like to express my sincere gratitude to Eunsol Choi, Ray Mooney, and Jonathan Berant for serving on my committee and providing valuable feedback on my dissertation. Thanks to Graham Neubig and Jonathan Berant, whose work has been a great source of inspiration for my research, for their support on my job search. I also thank Shixia Liu, my undergraduate advisor, for her continued guidance and mentorship after my graduation. Thanks to my internship mentors: Semih Yavuz, Yingbo Zhou, Ram Pasunuru, Srini Iyer, Ruoxi Sun, and Sercan Arik for their guidance.

I feel incredibly lucky to have my friends and fellow collaborators. Jocelyn Chen: thank you for all the papers we've co-authored and all the good times we've had together. I started my first NLP project working with Jocelyn, and we immediately started discussing (and fighting) about projects. We've worked together so well, and I'm looking forward to continuing our collaboration at our soon-to-be new school. Special thanks for putting up with all my complaints about research projects and life. I'd also like to thank all the members we overlapped with at TAUR and UT NLP. Jiacheng Xu: we had many interesting conversations on stressful topics and I learned many practical tips for surviving the PhD. Jifan Chen, Yasu Onoe, and Jialin Wu: chatting with you about all the random things has always been fun. Tanya Goyal and Kaj Bostrom: it is a pleasure to discuss with you on various topics. Fangcong Yin and Zeyu Liu: thank you for the chit-chats and the special Guandan training. I would like to thank the junior students with whom I collaborated: Prasann Singhal, Zayne Sprague, Weikai Yang, Yoonsang Lee, Fangcong Yin, and Zeyu Liu. It was fantastic to work with you. I would also like to thank my friends from the broader NLP community, Chenglei Si (special thanks for our many discussions), Jun Yan, Shiyang Li, Xinlu Zhang, Man Luo, Yunmo Chen, Wenhao Yu, and Ansong Ni. Thanks to Lemeng Wu, Zhen Chen, and Huancheng Chen for hanging out together in Austin. Thanks to Yu Fang and Jiaoying Mu for being my friend since undergrad. Thanks to Julia, the sweetest cat in the world, for her companionship and to an old friend for all the free therapy during my difficult times.

Finally, I would like to thank my parents for their unconditional love and support. Without them, I would not have been able to come this far.

Abstract

Steering Textual Reasoning with Explanations

Xi Ye, PhD
The University of Texas at Austin, 2024

SUPERVISOR: Greg Durret

Recent breakthroughs in pretraining have significantly extended the boundaries of language models' (LMs') potential applications, partially because of their increased ability to do complex reasoning. However, LMs have well-documented reasoning failures, such as hallucinations and inability to systematically generalize. In this dissertation, we aim to steer LMs in reliably performing textual reasoning, with a particular focus on leveraging explanations. We describe our work on steering textual reasoning with explanations in two paradigms: 1) intervening on model predictions post-hoc by using explanations from LMs to verify their predictions, and 2) teaching LMs to reason by demonstrating the reasoning process of solving reasoning tasks to them.

We first introduce how to leverage post-hoc explanations for intervening on model predictions. Past work has attempted to use post-hoc explanations for interpreting and debugging model behavior but often heavily relies on human effort. We focus instead on automating the process of using explanations to improve model predictions. Through a case study on QA models, we show that pairwise interaction-based explanation techniques align well with QA model behavior on counterfactuals, highlighting the connection between explanations and model behavior. This motivates us to introduce a framework for automatically assessing the robustness of black-box models using explanations. The framework first extracts features to describe the "reasoning process" disclosed by the explanations, and then

uses a trained verifier to judge the reliability of predictions based on these features. Using our framework, we successfully improve two classes of models on diverse tasks spanning QA, NLI, and commonsense reasoning: BERT-based models, improved using attributions, and GPT-3-based in-context learning, using free-text explanations.

We further study how to use explanations for teaching LMs to reason, especially free-text explanations for large language models (LLMs). We show that the performance of LLMs on downstream tasks is sensitive to the choice of explanations (among varied possible explanations) provided to them. We therefore propose methods for constructing effective explanations for LLMs. We introduce an approach that automatically optimizes explanations using unlabeled data, reducing the requirement of heavy manual prompt engineering. We also propose a framework that uses declarative formal specifications as explanations and employs an SMT solver to amend the limited planning capabilities of LLMs, which scales LLMs to handle problems requiring significantly deeper reasoning depth.

Lastly, we outline future directions for further enhancing LLMs to better aid humans in challenging real-world applications demanding deep reasoning.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Humans are capable of reasoning about information presented in language, when seeking conclusions from news articles, planning out schedules for work, or just making day-to-day small talk. Being able to reason over texts is also a key pursuit in natural language processing (NLP). It is crucial in a diversity of NLP applications, spanning from reading comprehension, scientific article summarization, to coding assistant. However, reasoning as an inherent skill to humans has been a longstanding challenge for NLP systems. Traditional NLP systems in the pre-pretraining era have struggled in scaling to even shallow reasoning tasks such as answering straightforward questions over text (Rajpurkar et al., 2016) or solving practical math problems involving only one or two steps (Patel et al., 2021).

Thanks to breakthroughs in pretraining, language models (LMs) have made remarkable progress in reasoning capabilities. Finetuned BERT-based models are now able to answer multi-hop questions (Yang et al., 2018). Large language models (LLMs) can even learn to solve challenging math questions (Hendrycks et al., 2021) from just a few examples (Lewkowycz et al., 2022). Despite the promising progress, LMs' reasoning capabilities are not robust. LMs often suffer from severe performance degradation under various adversarial attacks (Jia and Liang, 2017; Jiang and Bansal, 2019; Wallace et al., 2019). LMs are also susceptible to spurious correlations in the training data (Min et al., 2019; Chen and Durrett, 2019; McCoy et al., 2019; Bastings et al., 2021), which lead to generalization failures. At the same time, the internal reasoning process of LMs often lacks interpretability, making it hard to identify and debug the issues in the process.

Our work aims to leverage explanations to enable robust and complex textual reasoning with LMs. Explanations can be beneficial in multiple aspects. Most directly, faithful explanations can aid humans in **interpreting** model predictions, which facilitates identifying potential issues of models' reasoning process. In addition, explanations can be used for **verifying** model predictions, as they can be used to assess the reliability of a prediction. Furthermore, explanations can be used for **teaching** the reasoning process, when we are able to supervise LMs in following human-annotated correct reasoning process as specified in explanations.

Past work on using explanations before the widespread adoption of LLMs largely focuses on interpreting model predictions (Sundararajan et al., 2017; Ribeiro et al., 2016; Lundberg and Lee, 2017; Guan et al., 2019; De Cao et al., 2020). There has been some effort contributed to leveraging explanations for debugging models (Bastings et al., 2021; Adebayo et al., 2022) or incorporating explanations as additional supervision (Zaidan et al., 2007; Hancock et al., 2018; Rajani et al., 2019; Dua et al., 2020), but these typically rely on heavy human intervention, requiring either domain expert involvement (Adebayo et al., 2022) or substantial high-quality human annotations (Zaidan et al., 2007; Rajani et al., 2019; Press et al., 2022). More recent work on LLMs finds that including a few explanations for the in-context examples in prompts can improve LLMs' reasoning capabilities (Brown et al., 2020; Wei et al., 2022c). Nevertheless, different explanations for the same set of in-context examples can lead to widely varying downstream performance; prior research typically engineer explanation-infused prompts manually for various reasoning tasks (Zhou et al., 2022a; Press et al., 2022; Zhou et al., 2022b; Jung et al., 2022). Moreover, even with explanations, LLMs still exhibit reasoning failures such as hallucinations (discussed in Chapter 4) and inability to systematically generalize (Dziri et al., 2023).

1.1 Contributions of This Dissertation

The bulk of this dissertation focuses on steering LMs to perform complex textual reasoning reliably using explanations. We aim to improve model robustness and performance on downstream tasks while minimizing the need for human involvement in inspecting or annotating explanations. We introduce two paradigms of using explanations: the first paradigm uses explanations from LMs to verify their predictions and intervene on the predictions accordingly; the second paradigm provides explanations for LMs to demonstrate the reasoning process for solving a task. Our exploration covers both encoder-decoder LMs (such as BERT, RoBERTa) and decoder-only LLMs (like GPT-3) across various textual reasoning tasks.

We first introduce how to leverage post-hoc explanations to intervene on model predictions. In **Chapter 2**, we present a case study on QA models to investigate the connection between explanations and model behavior. Given a base QA example and model prediction, we may hypothesize how model predictions would differ if we changed a part of the inputs. We can perturb that particular part in a meaningful way, yielding a set of realistic counterfactuals. The model predictions on the counterfactuals can be used to verify whether the hypothesis holds true. We evaluate several attribution methods, a prominent thrust of explanation techniques, based on whether they correctly attend to the perturbed part in a way that aligns with the model's true behavior. Our work find that pairwise interaction-based explanation techniques align well with QA model behavior on realistic counterfactuals.

In **Chapter 3**, we study the utility of explanations for verifying black-box models' predictions. We propose a framework that can leverage attributions for calibrating RoBERTa models (Liu et al., 2019). Our framework first extracts a set of features combining human intuition about the task with model attributions generated by black box interpretation techniques, then uses a simple calibrator, in the form of a classifier, to predict whether the base model was correct or not. We experiment with our method on two tasks, extractive question answering and natural language inference, covering adaptation from several pairs of domains with limited target-domain data. Our evaluation establishes that explanations are useful for verifying model predictions, boosting their performance in the selective prediction setting.

In **Chapter 4**, we investigate how explanations can be used for improving the performance of prompting large LM like GPT-3 with in-context learning. We study this question on two NLP tasks that involve reasoning over text, namely question answering and natural language inference. We test the performance of four LLMs on three textual reasoning datasets using prompts that include explanations in multiple different styles. For these tasks, we find that including explanations in the prompts generally only yields small to moderate accuracy improvements over standard few-show learning on most of these models. We further show that explanations generated by the LLMs may not entail the models' predictions nor be factually grounded in the input, even on simple tasks with extractive explanations. However, these flawed explanations can still be useful as a way to verify LLMs' predictions post-hoc using the framework we described above. Again, we train verifiers using automatically extracted scores that assess the reliability of explanations, allowing us to improve performance post-hoc across datasets.

We further study how to use explanations for teaching LMs to reason, especially using free-text explanations for large language models (LLMs). In **Chapter 5**, we show explanations that have not been "tuned" for a task, such as off-the-shelf explanations written by non-experts, may lead to mediocre performance. We therefore tackle the problem of how to optimize explanation-infused prompts in a black-box fashion. We first generate sets of candidate explanations for each example in the prompt using a leave-one-out scheme, then find an effective *combination* of these explanations with a two-stage framework. We first evaluate explanations for each in-context example *in isolation* according to two proxy metrics, log likelihood and accuracy on new examples. Then, we search over combinations of explanations to find one that yields high performance against a silver-labeled development set. Through evaluation across four datasets, we show our method can effectively improve prompts over crowdworker annotations and naive search strategies.

In **Chapter 6**, we propose a *satisfiability-aided language modeling* (SATLM) framework. Our framework uses formal explanations and symbolic solvers to amend the fundamental planning limitations of LLMs, which cannot be addressed by simply optimizing imperative chain-of-thought style explanations. By offloading the actual reasoning task to an automated theorem prover, our approach can guarantee the correctness of the answer with respect to the parsed specification and avoid planning errors in the solving process. We evaluate SATLM on a wide range of reasoning tasks and show that it consistently outperforms baselines that use imperative explanations. The organic combination of the LLM and the symbolic solver scales LLMs to handle problems requiring significantly deeper reasoning depth.

Finally, **Chapter 7** we briefly summarize our contributions and outline future directions for further enhancing LLMs to better aid humans in challenging real-world applications demanding deep reasoning.

Chapter 2: Connecting Attributions and QA Model Behavior[1]

2.1 Introduction

Interpreting the behavior of black-box neural models for NLP has garnered interest for its many possible benefits (Lipton, 2018). Numerous post-hoc explanation techniques have been proposed, including textual explanations (Hendricks et al., 2016) and token-level attributions (Ribeiro et al., 2016; Sundararajan et al., 2017; Guan et al., 2019; De Cao et al., 2020). These formats can be applied to many domains, including sentiment analysis (Guan et al., 2019; De Cao et al., 2020), visual recognition (Simonyan et al., 2014), and natural language inference (Camburu et al., 2018; Thorne et al., 2019). However, it is hard to evaluate whether these explanations are *faithful* to the computation of the original model (Wu and Mooney, 2019; Hase and Bansal, 2020; Wiegreffe et al., 2021; Jacovi and Goldberg, 2020), and they can even mislead users (Rudin, 2019). More critically, token attributions in particular do not have a consistent and meaningful *social attribution* (Miller, 2019; Jacovi and Goldberg, 2021): that is, when a user of the system looks at the explanation, they do not necessarily draw a correct conclusion from it, making it hard to use for downstream tasks.

Our focus in this chapter is to investigate whether explanations for reading comprehension are capable of indicating the **high-level** behavior of models. That is, rather than a vague conclusion like "this word was important," we want to draw a conclusion like "the model compared these two words to reach its decision;" this statement can be evaluated for faithfulness and it helps a user draw meaningful conclusions about how system behaves. We approach this evaluation from a perspective of simulatability (Hase and Bansal, 2020): can we predict how the system will behave on new or modified examples? Doing so for RC

[1] An early version of this chapter has been published in Ye et al. (2021b). Xi Ye is the first author of Ye et al. (2021b), where he developed the research idea with other author(s), implemented the code, designed and performed the experiments and analysis, and wrote the paper.

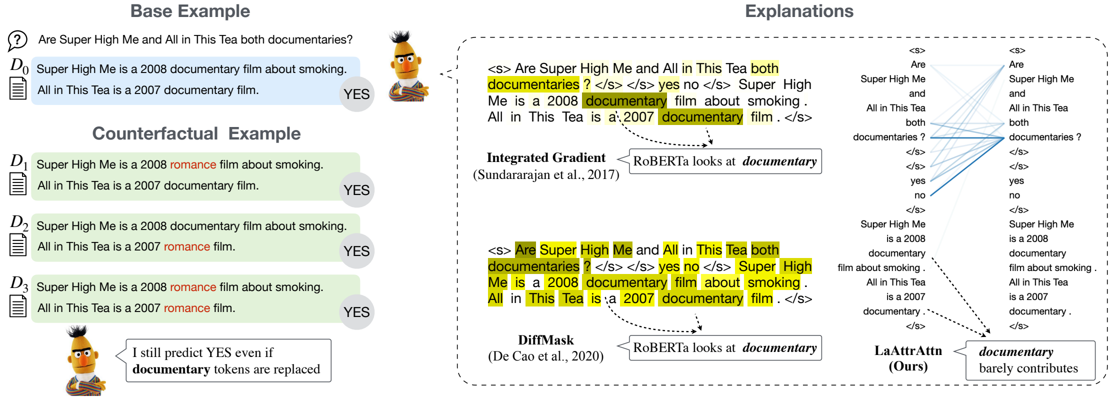

Base Example
Are Super High Me and All in This Tea both documentaries?
D_0
Super High Me is a 2008 documentary film about smoking.
All in This Tea is a 2007 documentary film.
YES
Counterfactual Example
D_1
Super High Me is a 2008 romance film about smoking.
All in This Tea is a 2007 documentary film.
YES
D_2
Super High Me is a 2008 documentary film about smoking.
All in This Tea is a 2007 romance film.
YES
D_3
Super High Me is a 2008 romance film about smoking.
All in This Tea is a 2007 romance film.
YES
I still predict YES even if documentary tokens are replaced
Explanations
<s> Are Super High Me and All in This Tea both documentaries ? </s> </s> yes no </s> Super High Me is a 2008 documentary film about smoking . All in This Tea is a 2007 documentary film . </s>
Integrated Gradient (Sundararajan et al., 2017)
RoBERTa looks at documentary
<s> Are Super High Me and All in This Tea both documentaries ? </s> </s> yes no </s> Super High Me is a 2008 documentary film about smoking . All in This Tea is a 2007 documentary film . </s>
DiffMask (De Cao et al., 2020)
RoBERTa looks at documentary
<s> Are Super High Me and All in This Tea both documentaries ? </s> </s> yes no </s> Super High Me is a 2008 documentary film about smoking . All in This Tea is a 2007 documentary . </s>
<s> Are Super High Me and All in This Tea both documentaries ? </s> </s> yes no </s> Super High Me is a 2008 documentary film about smoking . All in This Tea is a 2007 documentary . </s>
LaAttrAttn (Ours)
documentary barely contributes

Figure 2.1: A motivating example and explanations generated by several methods. We profile the model behaviors with the predictions on realistic counterfactual inputs, which suggests the model does not truly base its prediction on the two movies being documentaries. We can evaluate explanations by seeing whether they can be used in combination with heuristics to derive this same conclusion about model behavior.

models is challenging due to the complex nature of the task, which fundamentally involves a correspondence between a question and a supporting text context.

Our core technique is to assess how well various explanations can support or reject hypotheses about the model's behavior (i.e., simulate the model) on realistic counterfactuals, which are perturbations of original data points (Figure 2.1). These resemble several prior "stress tests" used to evaluate models, including counterfactual sets (Kaushik et al., 2020), contrast sets (Gardner et al., 2020), and checklists (Ribeiro et al., 2020). We first **semi-automatically** curate these sets to answer questions like: if different facts were shown in the context, how would the model behave? If different amounts of text or other incorrect paragraphs were retrieved by an upstream retrieval system, would the model still get the right answer? Then, given attributions from various techniques, can we *recover the answers to these questions* and give usable insights about the QA system?

We investigate two paradigms of explanation techniques, token attribution-based (Simonyan et al., 2014; Ribeiro et al., 2016; De Cao et al., 2020) and feature interaction-based (Tsang et al., 2020; Hao et al., 2020), which attribute decisions to sets of tokens or pairwise/higher-order interactions. We show that token-level attribution is not sufficient for analyzing QA, which naturally involves more complex reasoning over multiple clues.

For both techniques, we devise methods to bridge from these explanations to high-level conclusions about counterfactual behavior. This enables comparing different formats of explanations in a unified way.

We apply our methodology to automatically compare these attribution techniques on two types of questions from HOTPOTQA (Yang et al., 2018) and questions from adversarial SQUAD (Rajpurkar et al., 2016).

For each concrete high-level hypothesis we formulate, we automatically assess the extent to which our low-level explanation techniques can usefully produce the same answer as our counterfactuals. Our experimental results show moderate success of this approach overall, and that explanations in form of feature interactions better align with model behaviours. We further propose a modification to an existing interaction technique from Hao et al. (2020) and show improved performance on our datasets.

Our main contributions are: (1) We propose a new goal for attributions, namely automatically simulating model behavior on realistic counterfactuals. (2) We describe a technique for connecting low-level attributions (token-level or higher-order) with high-level model hypotheses. (3) We improve an attention-based pairwise attribution technique with a simple but effective fix, leading to strong empirical results. (4) We analyze a set of QA tasks and show that our approach can derive meaningful conclusions about counterfactuals on each.

2.2 Motivation

We start by going through a detailed example of how model attributions can be used for our proposed goal, and consequently how to use our methodology to compare several attribution techniques. Figure 2.1 shows an example of a multi-hop yes/no question from HotpotQA. The QA model correctly answers *yes* in this case. Given the original example, the explanations produced using INTGRAD (Sundararajan et al., 2017) and DIFFMASK (De Cao et al., 2020) (explained in Section 2.4) both assign high attribution scores to the two *documentary* tokens appearing in the context: a user of the system is likely to impute that the model is comparing these two values, as it's natural to assume this model is using the highlighted information correctly. By contrast, our pairwise attribution approach primarily attributes the prediction to interactions with the question, suggests the interaction related to *documentary* do not matter.

We manually curate a set of contrastive examples to test this hypothesis. If the model truly recognizes that both movies are documentaries, then replacing either or both of the *documentary* tokens with *romance* should change the prediction. To verify that, we perturb the original examples to obtain another three examples (left side of Figure 2.1). These four examples together form a contrastive local neighborhood (Ribeiro et al., 2016; Kaushik et al., 2020; Gardner et al., 2020) consisting of realistic counterfactuals.[2]

However, unlike what's suggested by the token attribution based techniques, the model always predicts "yes" for every example in the neighbourhood, casting doubt on whether the model is following the right reasoning process. Although the pairwise attribution seemed at first glance much less plausible than that generated by the other techniques, it was actually better from the perspective of simulating the model's behavior on these new examples.

Our main assumption in this chapter can be stated as follows: **an explanation should describe model behavior with respect to realistic counterfactuals, not just look plausible.** Past work has evaluated along plausibility criteria (Lei et al., 2016; Strout et al., 2019; Thorne et al., 2019), but as we see from this example, faithful explanations (Subramanian et al., 2020; Jacovi and Goldberg, 2020, 2021) are better aligned with our goal of simulatability. We argue that a good explanation is one that aligns with the model's high-level behaviors, and from which we can understand how the model generalizes to new data.

[2] One could argue that these counterfactuals are not entirely realistic: a romance film about smoking is fairly unlikely. Generating perfect counterfactuals is a very hard problem (Qin et al., 2019), requiring deep world knowledge of what scenarios make sense or what properties hold for certain entities. Nevertheless, we believe that these examples are realistic enough that robust models should still behave well on them.

Discussion: Realistic Counterfactuals Many counterfactual modifications are possible: past work has looked at injecting non-meaningful triggers (Wallace et al., 2019), deleting chunks of content (Ribeiro et al., 2016), or evaluating interpolated input points as in INTGRAD, all of which violate assumptions about the input distribution. In RC, masking out a fact in the question often turns the question into a nonsense one.[3] Focusing on realistic counterfactuals, by contrast, illuminates fundamental problems with our RC models' reasoning capabilities (Jia and Liang, 2017; Chen and Durrett, 2019; Min et al., 2019; Jiang and Bansal, 2019). This is the same motivation as that behind contrast sets (Gardner et al., 2020), but our work focuses on benchmarking explanations, not models themselves.

2.3 Behavior on Counterfactuals

We seek to formalize the reasoning we undertook on Figure 2.1. Using the model's explanation on a "base" data point, can we predict the model's behavior on the perturbed instances of that point?

Definitions Given an original example D_0 (e.g., the top example in Figure 2.1), we construct a set of perturbations based on $\{D_1, ..., D_k\}$ (e.g., the three counterfactual examples in Figure 2.1), which together with D_0 form a local neighborhood $\mathcal{D}$. These perturbations are realistic inputs derived from existing datasets or which we construct.

We formulate a hypothesis $\mathcal{H}$ about the neighborhood. In Figure 2.1, $\mathcal{H}$ is the question "is the model comparing the target properties?" (*documentary* in this case). Based on the model's behavior on the set $\mathcal{D}$, we can derive a high-level behavioral label z corresponding to the truth of $\mathcal{H}$. We form our local neighborhood to check the answer empirically and compute a ground truth for z. Since the model always predicts "yes" in this neighborhood, we label set $\mathcal{D}$ with $z = 0$ (the model is not comparing the properties). We label $\mathcal{D}$ as $z = 1$, when the model does predict "no" for some perturbations.

[3] The exception is in adversarial settings; however, many adversarial attacks do not draw on real-world threat models (Athalye et al., 2018), so we consider these less important.

Procedure Our approach is as follows:

1. Formulate a hypothesis $\mathcal{H}$ about the model

2. Collect realistic counterfactuals $\mathcal{D}$ to answer it empirically for some base examples

3. Use the explanation of each base example to predict z. That is, learn the mapping $D_0 \rightarrow z$ based on the explanation of D_0 so we can **simulate the model** on $\mathcal{D}$ without observing the perturbations.

Note that this third step *only* uses the explanation of the *base* data point: explanations should let us make conclusions about new counterfactuals without having to do inference on them.

Simulation In our experiments on HOTPOTQA and SQUAD, we compute a scalar factor f for each attribution representing the importance of a specific part of the inputs (e.g., the "documentary" tokens in Figure 2.1), which we believe should correlate with model predictions on the counterfactuals. If an attribution assigns higher importance to this information, it suggests that the model will actually change its behavior on these new examples.

Given this factor, we construct a simple classifier where we predict $z = 1$ if the factor f is above a threshold. We expect the factors extracted using better attribution methods should better indicate the model behavior. Hence, we evaluate the explanation using the **best simulation accuracy it can achieve** and the AUC score (S-ACC and S-AUC).[4]

Our evaluation resembles the human evaluation in Hase and Bansal (2020), which asks human raters to predict model's decision given an example together with its explanations and also reports simulatability. Our method differs in that (1) we predict the behavior on unseen counterfactuals given the explanation of a single base data point, and (2) we automatically extract a factor to predict model behavior instead of asking humans to do so.

[4] We do not collect large enough datasets to train a simulation model, but given larger collections of counterfactuals, this is another approach one could take.

2.4 Explanation Techniques

Compared to classification tasks like sentiment analysis, QA much more fundamentally involves interaction between input features, especially between a question and a context. This chapter will directly compare feature interaction explanations with token attribution techniques that are more common for other tasks.[5]

2.4.1 Token Attribution-Based

These techniques all return scores s_i for each token i in both the question and context that are fed into the QA system.

LIME (Ribeiro et al., 2016) and **SHAP** (Lundberg and Lee, 2017) both compute the attribution values for individual input features by using a linear model to locally approximate the model's predictions on a set of perturbed instances around the base data point. The attribution value for an individual input feature is the corresponding weight of the linear model. LIME and SHAP are different in the way of specifying instance weights used to train the linear model: LIME decides the weights heuristically, whereas SHAP specifies the weights according to Shapley values.

Integrated Gradient (INTGRAD) (Sundararajan et al., 2017) computes an attribution for each token by integrating the gradients of the prediction with respect to the token embeddings over the path from a baseline input (typically mask or pad tokens) towards the designated input. Although a common technique, recent work has raised concern about the effectiveness of INTGRAD methods for NLP tasks, as interpolated word embeddings do not correspond to real input values (Harbecke, 2021).

Differentiable Mask (DIFFMASK) (De Cao et al., 2020) learns to mask out a subsets of the input tokens for a given example while maintaining a distribution over answers as

[5] A potentially even more powerful format would be a program approximating the model's behavior, as has been explored in the context of reinforcement learning (Verma et al., 2018; Bastani et al., 2018). However, beyond limited versions of this (Ribeiro et al., 2018), prior work does not show how to effectively build this type of explanation for QA at this time.

close to the original distribution as possible. This mask is learned in a differentiable fashion, then a a shallow neural model (a linear layer) is trained to recognize which tokens to discard.

2.4.2 Feature Interaction-Based

These techniques all return scores s_i for each pair of tokens (i, j) in both the question and context that are fed into the QA system.

Archipelago (Tsang et al., 2020) measures non-additive feature interaction. Similar to DIFFMASK, ARCHIP is also implicitly based on unrealistic counterfactuals which remove tokens. Given a subset of tokens, ARCHIP defines the contribution of the interaction by the the prediction obtained from masking out all the other tokens, only leaving a very small fraction of the input. Applying this definition to a complex task like QA can result in a completely nonsensical input.

Attention Attribution (ATATTR) (Hao et al., 2020) uses attention specifically to derive pairwise explanations. However, it avoids the pitfalls of directly inspecting attention (Serrano and Smith, 2019; Wiegreffe and Pinter, 2019) by running an integrated gradients procedure over all the attention links within transformers, yielding attribution scores for each link. The attribution scores directly reflect the attribution of the particular attention links, making this model able to describe **pairwise interactions**.

Concretely, define the h-head attention matrix over input D with n tokens as $A = [A_1, ..., A_l]$, where $A_i \in \mathbb{R}^{h \times n \times n}$ is the attention scores for each layer. We can obtain the attribution score for each entry in the attention matrix A as:

$$\mathrm{ATTR}(A) = A \odot \int_{\alpha=0}^{1} \frac{\partial F(D, \alpha A)}{\partial A} d\alpha, \tag{2.1}$$

where $F(D, \alpha A)$ is the transformer model that takes as input the tokens and a matrix specifying the attention scores for each layer. We later sum up the attention attributions across all heads and layers to obtain the pairwise interaction between token (i, j), i.e., $s_{ij} = \sum_m \sum_n \mathrm{ATTR(A)_{mnij}}$.

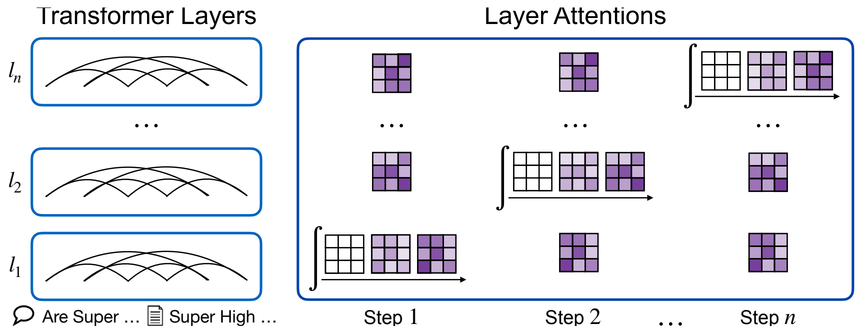


Figure 2.2: Steps of our Layer-wise Attention Attribution approach, where we only intervene a single layer at step. E.g., to compute the attribution of attentions at layer 2, we only intervene the attention matrix at that layer, and leave other attentions computed as usual.

2.4.3 Layer-wise Attention Attribution

We propose a new technique LATATTR to improve upon ATATTR for the RC setting. The ATATTR approach simultaneously increases all attention scores when computing the attribution, which could be problematic. Since the attention scores of higher layers are determined by the attention scores of lower layers, forcibly setting all the attention scores and computing gradients at the same time may distort the gradients for the lower level links and produce inaccurate attribution. When applying INTGRAD approach in other contexts, we typically assume the independence of input features (e.g., pixels of an image and tokens of an utterance), an assumption which does not hold here.

To address this issue, we propose a simple fix, namely applying the INTGRAD method layer-by-layer. As in Figure 2.2, to compute the attribution for attention links of layer i, we only change the attention scores at layer i:

$$\mathrm{ATTR}(A_i) = A_i \odot \int_{\alpha=0}^{1} \frac{\partial F_{/i}(D, \alpha A_i)}{\partial A_i} d\alpha. \tag{2.2}$$

$F_{/i}(D, \alpha A_i)$ denotes that we only intervene on the attention masks at layer i while leaving other attention masks computed naturally via the model. We pool to obtain the final attribution for pairwise interaction as $s_{ij} = \sum_m \sum_n \mathrm{ATTR(A)}_{\mathrm{mnij}}$.

This technique does not necessarily satisfy the Completeness axiom commonly used in this line of work (Sundararajan et al., 2017). Since our ultimate goal is a downstream

(a)	**Question:** Were Ulrich Walter and Léopold Eyharts both from Germany? **Context:** Léopold Eyharts (born April 28, 1957) is a Brigadier General in the French Air Force, an engineer and ESA astronaut. Prof. Dr. Ulrich Hans Walter (born February 9, 1954) is a German physicist/engineer and a former DFVLR astronaut. **Substitutes:** French, German
(b)	**Question:** Are the movies ”Monsters, Inc.” and ”Mary Poppins” both by the same company? **Context:** Mary Poppins is a 1964 American musical-fantasy film directed by Robert Stevenson and produced by Walt Disney, with songs written and composed by the Sherman Brothers. Monsters, Inc. is a 2001 American computer-animated comedy film produced by Pixar Animation Studios and distributed by Walt Disney Pictures. **Substitutes:** Walt Disney, Universal
(c)	**Question:** What was the father of Kasper Schmeichel voted to be by the IFFHS in 1992? **Context:** Peter Bolesław Schmeichel MBE (born 18 November 1963) is a Danish former professional footballer who was voted the IFFHS World's Best Goalkeeper in 1992 and 1993. Kasper Peter Schmeichel (born 5 November 1986) is a Danish professional footballer. He is the son of former Manchester United and Danish international goalkeeper Manuel Neuer. **AdvSent1:** Robert Lewandowski was voted to be the World's Best Striker in 1992. **AdvSent2:** Michael Jordan was voted the IFFHS best NBA player in 1992.

Figure 2.3: Examples (contexts are truncated for brevity) of our property annotations on Hotpot base data points. The top two are yes/no questions and the third is a bridge question.

empirical evaluation, we set aside any theoretical analysis of this technique for now.

2.5 Experiments

We assess whether the explanations can achieve our proposed goal following the setup in Section 2.3 on the HOTPOTQA dataset (Yang et al., 2018), and the SQUAD dataset (Rajpurkar et al., 2016), specifically leveraging examples from adversarial SQUAD (Jia and Liang, 2017).

2.5.1 Hotpot Yes-No Questions

We first study a subset of comparison yes/no questions, which is a challenging format despite the binary answer space (Clark et al., 2019). Typically, a yes-no comparison type question requires comparing the properties of two entities (Figure 2.1). We base our experiments on a ROBERTA (Liu et al., 2019) QA model achieving 77.2 F1 scores on the development set in the distractor setting, comparable to other strong ROBERTA-based models (Tu et al., 2020; Groeneveld et al., 2020).

Hypothesis & Counterfactuals The hypothesis $\mathcal{H}$ we investigate is as in Section 2.2: *the model compares the entities' properties as indicated by the question.* Most Hotpot Yes-No questions follow one of two templates: *Are A and B both __?* (Figure 2.3a), and *Are A and B of the same __?* (Figure 2.3b). We define the **property** tokens associated with each question as the tokens *in the context* that match the blank in the template; that is, the values of the property that A and B are being compared on. For example, in Figure 2.3a, *French* and *German* are the property tokens, as the property of interest is the national origin.

To construct a neighborhood for a base data point, we take the following steps: 1) manually extract the property tokens in the context 2) replace the property token with two substitutes, forming a set of four counterfactuals exhibiting nonidentical ground truths. When the properties associated with the two entities differ from each other, we directly use the properties extracted as the substitutes (Figure 2.3a); otherwise we add a new property candidate that is of the same class (Figure 2.3b).

We set $z = 0$ (the hypothesis does not hold) if for each perturbed example $D_i \in \mathcal{D}$, the model predicts the same answer as for the original example, indicating a failure to compare the properties. We set $z = 1$ if the model's prediction *does* change. The authors annotate perturbations for 50 ($\mathcal{D}$, z) randomly selected pairs in total, forming a total of 200 counterfactual instances. Full counterfactual set can be found in supplementary materials.

Connecting Explanation and Hypothesis To make a judgment about z, we extract a factor f based on the importance of a set of property tokens P. For token attribution-based methods, we define f as the sum of the attribution s_i of each token in P: $\sum_{i \in P} s_i$. For feature interaction-based methods producing pairwise attribution s_{ij}, we compute f by pooling the scores of all the interaction related to the property tokens, i.e., $\sum_{i \in P \vee j \in P} s_{ij}$.

Now we predict $z = 1$ if the factor f is above a threshold, and evaluate the capability of the factor in indicating the model high-level behavior using the best simulation accuracy it can achieve (S-ACC) and AUC score (S-AUC).[6]

[6] Note that for different attribution methods, the thresholds are different and set to achieve the best accuracy.

Approach	Yes-No		Bridge	
	S-ACC	S-AUC	S-ACC	S-AUC
Majority	52.0	–	56.0	–
Conf	64.0	49.8	66.0	65.9
Lime	72.0	73.6	74.0	71.4
Shap	72.0	70.5	76.0	75.0
IntGrad	72.0	75.2	72.0	77.9
DiffMask	66.0	60.2	68.0	62.3
Archip	56.0	53.2	62.0	57.5
AtAttr	66.0	63.6	72.0	79.1
LatAttr	**84.0**	**87.9**	**78.0**	**81.7**

Table 2.1: Results on HotpotQA Yes-No type and Bridge questions. Our approach can better predict the model behavior on realistic counterfactuals, surpassing token attribution methods.

Results First, we show that using attributions can indeed help predict the model's behavior. In Table 2.1, our approach (LatAttr) is the best, achieving a simulation accuracy of 84%. That is, with a properly set threshold, we can successfully predict whether the model predictions change when perturbing the properties in the original example 84% of the time. The attributions therefore give us the ability to simulate our model's behavior better than the other methods here. Our approach also improves substantially over the vanilla AtAttr method.

Token attribution based approaches obtain an accuracy around 72%. This indicates token attribution based methods are not effective in the HotpotQA setting which engages with interaction between tokens more intensively.

In this setting, DiffMask performs poorly typically because it assigns high attribution to many tokens, since it determines which tokens need to be kept rather than distinguishing fine-grained importance (examples in supplementary materials). It's possible that other heuristics or models learned on large numbers of perturbations could more meaningfully extract predictions from this technique.

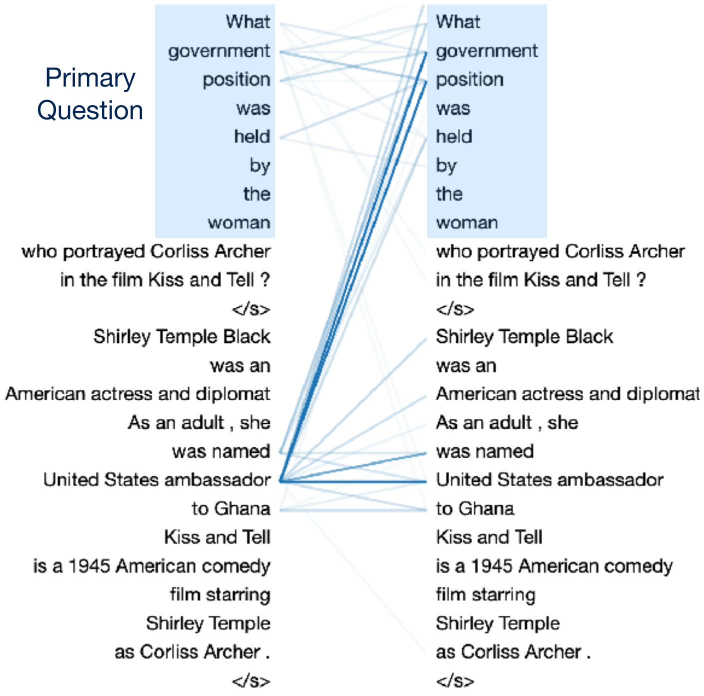


Figure 2.4: Explanations generated by our approach for a bridge type question from HotpotQA. The prediction can mostly be attributed to the primary question, indicating the model is taking the reasoning shortcut, and the prediction is flipped with an adversarial attack.

2.5.2 Hotpot Bridge Questions

We also evaluate the explanation approaches on so-called bridge questions on the HotpotQA dataset, described in Yang et al. (2018). Figure 2.4 shows a example explanation of a bridge problem. From the attribution scores we find the most salient connection is between the span “what government position” in the the question and the span “United States Ambassador” in the context. This attribution directly highlights the reasoning shortcut (Jia and Liang, 2017; Chen and Durrett, 2019; Min et al., 2019; Jiang and Bansal, 2019) the model is using, where it disregards the second part of the question. If we inject an additional sentence *“Hillary Clinton is an American politician, who served as the United States secretary of the state from 2009 to 2013”*, into the context, the model will be misled and predict “United States secretary” as the new answer. This sentence could easily have been part of another document retrieved in the retrieval stage, so we consider its inclusion to be a realistic counterfactual.

We further define the *primary* question, i.e., the primary part (containing wh-words) of the entire question. (E.g., “What government position is held by the woman” in Figure 2.4), following the decomposition principle from Min et al. (2019).

Hypothesis & Counterfactuals The hypothesis $\mathcal{H}$ we investigate is: *the model is using correct reasoning and not a shortcut driven by the primary question part.*

We construct counterfactuals following the same idea applied in our example. We view bridge questions as consisting of two single hop questions, the primary part and the secondary part. The primary part is the main body of the question, whereas the secondary part is usually a clause used to link the bridge entity (Min et al., 2019). For a given question, we add an adversarial sentence based on the primary part of the question so as to alter the model prediction. The added adversarial sentence contains context leading to a spurious answer to only the primary question, but does not change the gold answer. We do this twice, yielding a set $\mathcal{D} = \{D_0, D_1, D_2\}$ consisting of the base example and two perturbations. We define the label of D to be $z = 0$ in the case that model’s prediction does change when being attacked, and $z = 1$ otherwise. We show one example in Figure 2.3c. More examples and the full counterfactual set can be found in supplementary materials.

We randomly sample 50 base data points from the development set and two authors each write an adversarial sentence, giving 150 data points total.

Connecting Explanation and Hypothesis For this setting, we use a factor describing the importance of the primary question normalized by the importance of the entire question. Namely, let $P = \{p_i\}$ be the set of tokens in the primary questions, and $Q = \{q_i\}$ be the set of tokens in the entire question. We define the factor f as the the importance of P normalized by the importance of Q, where the importance calculation is the same as in Section 2.5.1. A higher factor means it is more heavily relying only on the primary question and hence a better chance of being attacked.

Results According to the simulation AUC scores in Table 2.1, feature interaction based techniques again outperform token attribution approaches. Our approach achieves a stimulation accuracy of 78%, substantially higher than any other results.

2.5.3 SQuAD Adversarial

Hypothesis & Counterfactuals Our hypothesis $\mathcal{H}$ is: *the model can resist adversarial attacks of the addSent variety* (Jia and Liang, 2017). For each of the original examples D_0 from a portion of the SQUAD-ADV development set, Jia and Liang (2017) creates 5 adversarial attacks, which are paraphrased and filtered by Turkers to give 0 to 5 valid attacks for each example, yielding our set $\mathcal{D}$. We define the label of $\mathcal{D}$ to be $z = 1$ if the model resists all the adversarial attacks posed on D_0 (i.e., predictions for D are the same). To ensure the behavior is more precisely profiled by the counterfactuals, we only keep the base examples with more than 3 valid attacks, resulting in a total number of 276 $(\mathcal{D}, z)$ pair (1,506 data points).

Connecting Explanation and Hypothesis We use a factor p indicating the importance of the essential keywords extracted from the question using POS tags (proper nouns and numbers). E.g., for the question "What Florida stadium was considered for Super Bowl 50", we extract "Florida", "Super Bowl" , and "50". If the model considers all the essential keywords mentioned in the question, it should not be fooled by distractors with irrelevant information. We show a set of illustrative examples in supplementary materials. We compute the importance scores in the same way described in Section 2.5.1.

In addition to the scores provided by various explanation techniques, we also use the model's confidence on the original prediction as a baseline.

Results We show results in Table 2.2. The best approaches (ATATTR and LATATTR) can achieve a simulation accuracy around 70%, 10% above the performance based on confidence. This shows the model is indeed over-confident in its prediction; our assumption about the

Approach	S-ACC	S-AUC
MAJORITY	52.1	–
CONF	58.3	57.8
LIME	67.7	68.3
SHAP	65.9	68.3
INTGRAD	61.6	61.1
DIFFMASK	57.6	53.6
ARCHIP	58.6	56.2
ATATTR	68.4	**72.5**
LATATTR	**70.0**	72.1

Table 2.2: Simulation Accuracy and AUC scores for the SQuAD adversarial setting, assessing whether model changes its prediction on an example when attacked.

robustness together with our technique can successfully expose the vulnerability in some of the model predictions.

There is room to improve on these results; our simple heuristic cannot perfectly connect the explanations to the model behavior in all cases. We note that there are other orthogonal approaches (Kamath et al., 2020) to calibrate the confidence of QA models' predictions by looking at statistics of the adversarial examples; here, our judgment is made purely based on the *original* example, and does not exploit learning to refine our heuristic.

2.5.4 Discussion and Limitations

We show that feature attributions can reveal known dataset biases and reasoning shortcuts in HotpotQA without having to perform a detailed manual analysis. This confirms the suitability of our attribution methods for at least this use case: model designers can look at them, either manually or automatically, and determine how robust the model is going to be when faced with counterfactuals.

Our analysis also highlights limitations of current explanation techniques. We experimented with other counterfactuals by permuting the order of the paragraphs in the context, which often gave rise to different predictions. We believe the model prediction was in these cases impacted by biases in positional embeddings (e.g., the answer tends to occur in the first retrieved paragraph), which cannot be indicated by current attribution methods. We believe this is a useful avenue for future investigation. By first thinking about what kind of counterfactuals and what kind of behaviours we want to explain, we can motivate the development of new explanation techniques to serve these needs.

2.6 Related Work

We focus on several prominent token attribution techniques, but there are other related methods as well, including Shapley Values (Štrumbelj and Kononenko, 2014; Lundberg and Lee, 2017), contextual decomposition (Jin et al., 2020), and hierarchical explanations (Chen et al., 2020). These formats can also be evaluated using our framework if being connected with model behavior with proper heuristic. Other work explores "concept-based" explanations (Mu and Andreas, 2020; Bau et al., 2017; Yeh et al., 2019). These provide another pathway towards building explanations of high-level behavior; however, they have been explored primarily for image recognition tasks and cannot be directly applied to QA, where defining these sorts of "concepts" is challenging.

Probing techniques aim to discover what intermediate representations have been learned in neural models (Tenney et al., 2019; Conneau et al., 2018; Hewitt and Liang, 2019; Voita and Titov, 2020). Internal representations could potentially be used to predict behavior on contrast sets similar to the work of this chapter; however, this cannot be done heuristically and larger datasets are needed to explore this.

Other work considering how to evaluate explanations is primarily based on how explanations can assist humans in predicting model decisions for a given example (Doshi-Velez and Kim, 2017; Chandrasekaran et al., 2018; Nguyen, 2018; Hase and Bansal, 2020); We are the first to consider building contrast sets for this. Similar ideas have been used in other contexts (Kaushik et al., 2020; Gardner et al., 2020) but our work focuses on evaluation of explanations rather than general model evaluation.

2.7 Conclusion

We have presented a new methodology of using explanations for understanding model behavior on realistic counterfactuals. We show explanations can indeed indicate model behavior, and therefore we can compare explanations to understand which ones truly give us insight about high-level model behavior. Feature interaction-based techniques perform the best in our analysis, especially our LATATTR method; we believe that this could be a useful evaluation paradigm if extended and formalized across a range of tasks.

Chapter 3: Calibrating Black-box Models Using Explanations[1]

3.1 Introduction

In Chapter 2, we have established the connection between explanations and model behavior, showcasing that explanations produced by a proper technique can provide useful information for indicating model behavior. In this chapter, we show how such a connection can be used for building a framework that can automatically assess model robustness based on explanations.

We study how post-hoc explanations can be used for improving black-box models, which are more and more prevalent throughout the Internet. NLP models are showing increasingly promising performance on real-world tasks, leading to their deployment at scale for translation, sentiment analysis, and question answering. These models are sometimes used as black boxes, especially if they are only available as a service through APIs[2] or if end users do not have the resources to fine-tune the models themselves. This black-box nature poses a challenge when users try to deploy models on a new domain that diverges from the training domain, usually resulting in performance degradation.

We investigate the task of domain adaptation of black box models: given a black box model and a small number of examples from a new domain, how can we improve the model's generalization performance on the new domain? In this setting, note that we are not able to update the model parameters, which makes transfer and few-shot learning techniques inapplicable. However, we can still make the model more effective in practice by learning a *calibrator*, or a separate model to make a binary decision of whether the black box model

[1] An early version of this chapter has been published in Ye and Durrett (2022a). Xi Ye is the first author of Ye and Durrett (2022a), where he developed the research idea with other author(s), implemented the framework, designed and performed the experiments and analysis, and wrote the paper.

[2] Google Translate, the Perspective API `https://perspectiveapi.com/`, and MonkeyLearn `https://monkeylearn.com/monkeylearn-api/` being three examples.

is likely to be correct or not on a given instance. While not fully addressing the domain adaptation problem, calibrating the model can make it more useful in practice, as we can recognize when it is likely to make mistakes (Guo et al., 2017; Kamath et al., 2020; Desai and Durrett, 2020) and modify our deployment strategy accordingly.

This chapter explores how explanations can help address this task. We leverage black box feature attribution techniques (Ribeiro et al., 2016; Lundberg and Lee, 2017) to identify key input features the model is leveraging, even without access to model internal representations. As shown in Figure 3.1, we perform calibration by connecting model interpretations with hand-crafted heuristics to extract a set of features describing the "reasoning" of the model. For the question answering setting depicted in the figure, answers turn out to be more reliable when the tokens of a particular set of tags (e.g., proper nouns) in the question are strongly considered. We extract a set of features describing the attribution values of different tags. Using a small number of examples in the target domain, we can train a simple calibrator for the black box model.

Our approach is closely related to the recent line of work on model behavior and explanations. Chandrasekaran et al. (2018); Hase and Bansal (2020) shows explanations can help users predict model decisions in some ways and Chapter 2 show how these explanations can be semi-automatically connected to model behavior based on manually crafted heuristics. Our approach goes further by using a model to learn these heuristics, instead of handcrafting them or having a human inspect the explanations.

We test whether our method can improve model generalization performance on two tasks: extractive question answering (QA) and natural language inference (NLI). We construct generalization settings for 5 pairs of source and target domains across the two tasks. Compared to existing baselines (Kamath et al., 2020) and our own ablations, we find explanations are indeed helpful for this task, successfully improving calibrator performance among all pairs. We even find settings where explanation-based calibrators outperform fine-tuning the model on target domain data, which assumes glass-box access to the model's parameters. Our analysis further demonstrates generalization of the calibrator models

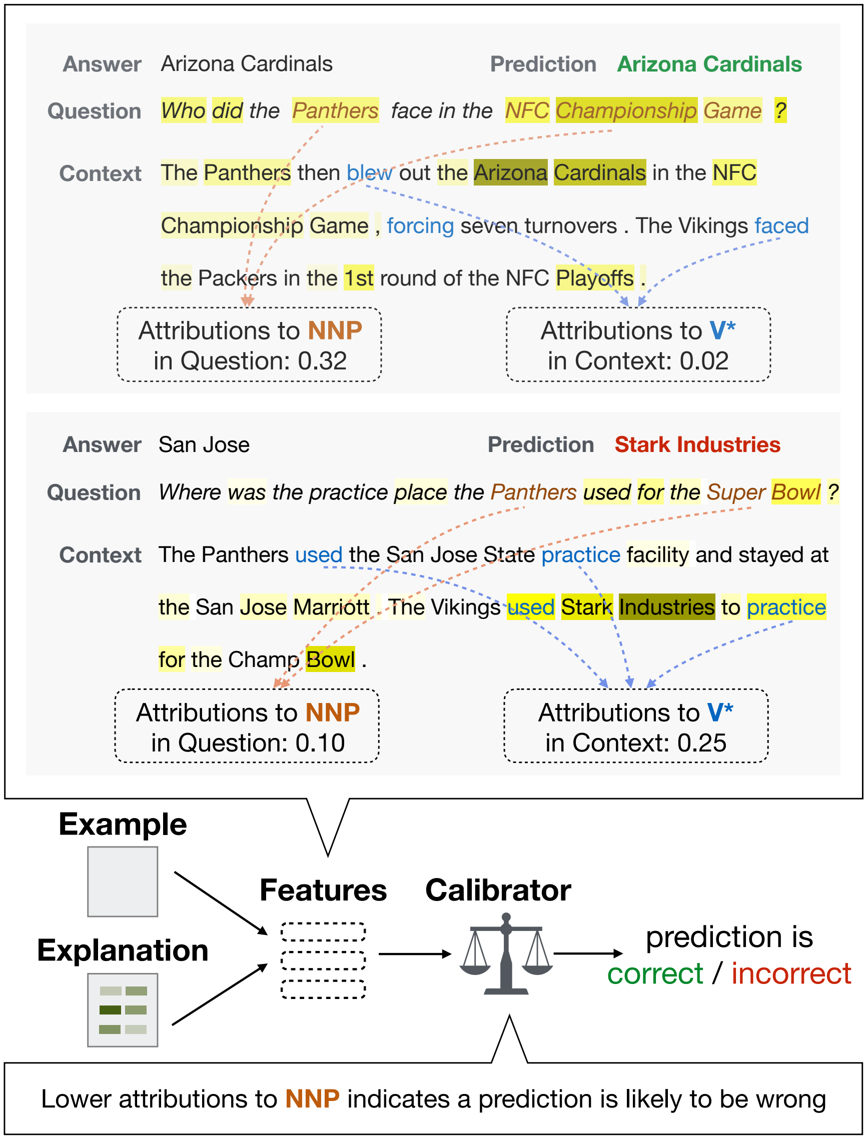

Answer Arizona Cardinals
Prediction Arizona Cardinals
Question Who did the Panthers face in the NFC Championship Game ?
Context The Panthers then blew out the Arizona Cardinals in the NFC Championship Game , forcing seven turnovers . The Vikings faced the Packers in the 1st round of the NFC Playoffs .
Attributions to NNP in Question: 0.32
Attributions to V* in Context: 0.02
Answer San Jose
Prediction Stark Industries
Question Where was the practice place the Panthers used for the Super Bowl ?
Context The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott . The Vikings used Stark Industries to practice for the Champ Bowl .
Attributions to NNP in Question: 0.10
Attributions to V* in Context: 0.25
Example
Explanation
Features
Calibrator
prediction is correct / incorrect
Lower attributions to NNP indicates a prediction is likely to be wrong

Figure 3.1: Calibrator pipeline and examples from the SQUAD-ADV dataset. A ROBERTA model trained on SQUAD is correct on the first example but incorrect on the second. Features that inspect attribution values produced by LIME can differentiate these two on the basis of attributions to NNP in the question and V* in the context. A calibrator using these features can predict whether the original model was right or wrong.

themselves: our calibrator trained on one domain can transfer to another new domain in some cases. Moreover, our calibrator can also substantially improves model performance in the Selective QA setting.

3.2 Using Explanations for Black Box Model Calibration

Let $x = x_1, x_2, ..., x_n$ be a set of input tokens and $\hat{y} = f(x)$ be a prediction from our black box model under consideration. Our task in calibration[3] is to **assess whether the model prediction on** x **matches its ground truth** y. We represent this with the variable t, i.e., $t \triangleq \mathbb{1}\{f(x) = y\}$.

We explore various calibrator models to perform this task, with our main focus being on calibrator models that leverage explanations in the form of *feature attribution*. Specifically, an explanation ϕ for the input x assigns an attribution score ϕ_i for each input token x_i, which represents the importance of that token. Next, we extract features $u(x, \phi)$ depending on the input and explanation, and use the features to learn a calibrator $c : u(x, \phi) \to t$ for predicting whether a prediction is valid. We compare against baselines that do not use explanations in order to answer the core question posed by our paper's title.

Our evaluation focuses on binary calibration, or classifying whether a model's initial prediction is correct. Following recent work in this setting (Kamath et al., 2020), we particularly focus on domain transfer settings where models make frequent mistakes. A good calibrator can identify instances where the model has likely made a mistake, so we can return a null response to the user instead of an incorrect one.

In the remainder of this section, we'll first introduce how we generate the explanations and then how to extract the features u for the input x.

[3] We follow Kamath et al. (2020) in treating calibration as a binary classification task. Devising a good classifier is connected to the goal of accurate estimation of posterior probabilities that calibration has more historically referred to (Guo et al., 2017), but our evaluation focuses on binary accuracy rather than real-valued probabilities.

3.2.1 Generating Explanations

Since we are calibrating black box models, we adopt LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017) for generating explanations for models instead of other techniques that require access to the model details (e.g., integrated gradients (Sundararajan et al., 2017)).

The rest of this chapter only relies on LIME and SHAP to map an input sequence x and a model prediction y to a set of importance weights ϕ. We will briefly summarize the unified framework shared by both methods, and refer readers to the respective papers for additional details.

LIME and SHAP generate *local explanations* by approximating the model's predictions on a set of perturbations around the base data point x. In this setting, a perturbation x' with respect to x is a simplified input where some of the input tokens are absent (replaced with a `<mask>` token). Let $z = z_1, z_2, ..., z_n$ be a binary vector with each z_i indicating whether x_i is present (using value 1) or absent (using value 0), and $h_x(z)$ be the function that maps z back to the simplified input x'. Both methods seek to learn a local linear classifier g on z which matches the prediction of original model f by minimizing:

$$g(z) = \phi_0 + \sum_{i=1}^{n} \phi_i z_i$$
$$\xi = \arg\min_{g} \sum_{z \in Z} \pi_x(z)[f(h_x(z)) - g(z)]^2 + \Omega(g)$$

where π_x is a local kernel assigning weight to each perturbation z, and Ω is the L2 regularizer over the model complexity. The learned feature weight ϕ_i for each z_i then represents the additive attribution (Lundberg and Lee, 2017) of each individual token x_i. LIME and SHAP differ in the choice of the local kernel π_x. Please refer to the supplementary materials for details of the kernel.

3.2.2 Extracting Features by Combining Explanations and Heuristics

Armed with these explanations, we now wish to connect the explanations to *the reasoning we expect from the task*: if the model is behaving as we expect, it may be better calibrated. A human might look at the attributions of some important features and decide whether the model is trustworthy in a similar fashion (Doshi-Velez and Kim, 2017). Chapter 2 has explored such a technique to compare explanation techniques. Past work also explored running studies with human users on this task (Chandrasekaran et al., 2018; Hase and Bansal, 2020).

Our method in this chapter automates this process by learning what properties of explanations are important. We first assign each token x_i with one or more human-understandable **properties** $V(x_i) = \{v_j\}_{j=1}^{m_i}$. Each property $v_j \in \mathcal{V}$ is an element in the property space, which includes indicators like POS tags and is used to describe an aspect of x_i whose importance might correlate with the model's robustness. We conjoin these properties with aspects of the explanation to render our calibration judgment. Figure 3.1 shows examples of properties such as whether a token is a proper noun (NNP).

We now construct the feature set for the prediction made on x. For every property $v \in \mathcal{V}$, we extract a single feature $F(v, x, \phi)$ by aggregating the attributions of the tokens associated with v:

$$F(v, x, \phi) = \sum_{i=1}^{n} \sum_{\bar{v} \in V(x_i)} \mathbb{1}\{\bar{v} = v\}\phi_i$$

where $\mathbb{1}$ is the indicator function, and ϕ_i is the attribution value. An individual feature represents the total attribution with respect to property v when the model is making the predictions for x. The complete feature set u for x, given as $u = \{F(v, x, \phi)\}_{v \in \mathcal{V}}$, summarizes model rationales from the perspective of the properties in $\mathcal{V}$.

Properties We use several types of heuristic properties for calibrating QA and NLI models.

Segments of the Input (QA and NLI): In both of our tasks, an input sequence can naturally be decomposed into two parts, namely a question and a context (QA) or a premise and a hypothesis (NLI). We assign each token with the corresponding segment name, which yields features like `Attributions to Question`.

POS Tags (QA and NLI): We use tags from the English Penn Treebank (Marcus et al., 1993) to implement a group of properties. We hypothesize that tokens of some specific tags should be more important, like proper nouns in the questions of the QA tasks. If a model fails to consider proper nouns of a QA pair, it is more likely to make incorrect predictions.

Overlapping Words (NLI): Word overlap between a premise and a hypothesis strongly affects neural models' predictions (McCoy et al., 2019). We assign each token with the `Overlapping` property if a token appears in both the premise and the hypothesis, or `Non-Overlapping` otherwise.

Conjunction of Groups: We can further produce higher-level properties by taking the Cartesian product of two or more groups. We conjoin `Segment` and `Pos-Tags`, which yields higher-level features like `Attributions to NNP in Question`. Such a feature aggregates attributions of tokens that are tagged with `NNP` and also required to be in the question (marked with orange).

3.2.3 Calibrator Model

We train the calibrator on a small number of samples in our target domain. Each sample is labeled using the prediction of the original model compared to the ground truth. Using our feature set $F(v, x, \phi)$, we learn a random forest classifier, shown to be effective for a similar data-limited setting in Kamath et al. (2020), to predict t (whether the prediction is correct). This classifier returns a score, which overrides the model's original confidence score with respect to that prediction.

In Section 3.4, we discuss several baselines for our approach. As we vary the features used by the model, all the other details of the classifier and setup remain the same.

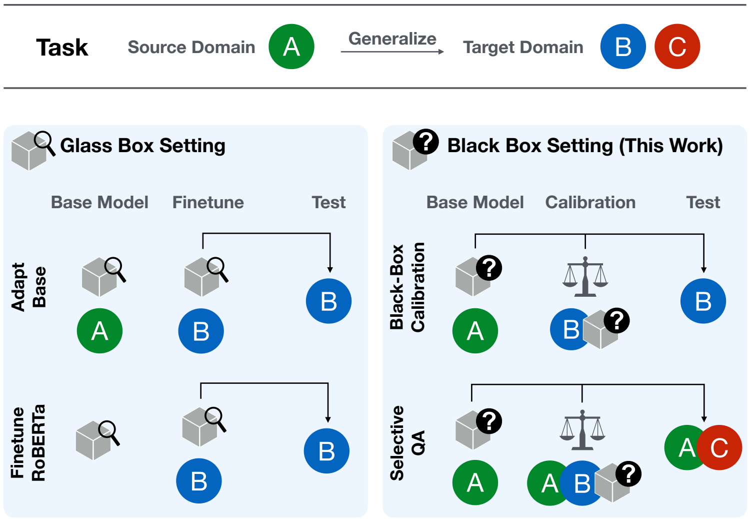


Figure 3.2: Illustration of different settings in the experiments. In black box settings, a calibrator is trained for improving model performance on OOD data; in glass box settings, the model is finetuned on OOD data from a base model or vanilla RoBERTa LM model.

3.3 Tasks and Datasets

Our task setup involves transferring from a source domain/task A to a target domain/task B. Figure 3.2 shows the data conditions we operate in. Our primary experiments focus on using our features to either calibrate or selectively answer in the black box setting (right side in Figure 3.2). In this setting, we have a black box model trained on a source domain A and a small amount of data from the target domain B. Our task is to train a calibrator using data from domain B to identify instances where the model potentially fails in the large unseen test data in domain B. We contrast this black box setting with glass box settings (left side in Figure 3.2), where we directly have access to the model parameters and can fine-tune on domain B or train on B from scratch.

English Question Answering We experiment with domain transfer from SQuAD (Rajpurkar et al., 2016) to three different settings: SQuAD-Adv (Jia and Liang, 2017), HotpotQA (Yang et al., 2018), and TriviaQA (Joshi et al., 2017).

SQuAD-Adv is an adversarial setting based on SQuAD, which constructs adversarial

QA examples based on SQUAD by appending a distractor sentence at the end of each example's context. The added sentence contains a spurious answer and usually has high surface overlapping with the question so as to fool the model. We use the ADDSENT setting from Jia and Liang (2017).

Similar to SQUAD, HOTPOTQA also contains passages extracted from Wikipedia, but HOTPOTQA asks questions requiring multiple reasoning steps, although not all questions do (Chen and Durrett, 2019). TRIVIAQA is collected from Web snippets, which present a different distribution of questions and passages than SQUAD. For HOTPOTQA and TRIVIAQA, we directly use the pre-processed version of dataset from the MRQA Shared Task (Fisch et al., 2019).

English NLI For the task of NLI, we transfer a model trained on MNLI (Williams et al., 2018) to MRPC (Dolan and Brockett, 2005) and QNLI (Wang et al., 2019), similar to the settings in Ma et al. (2019). QNLI contains a question and context sentence pair from SQUAD, and the task is to verify whether a sentence contains the answer to the paired question. MRPC is a paraphrase detection dataset presenting a binary classification task to decide whether two sentences are paraphrases of one another. Note that generalization from MNLI to QNLI or MRPC not only introduces shift in terms of the distribution of the input text, but in terms of the nature of the task itself, since QNLI and MRPC aren't strictly NLI tasks despite sharing some similarity. Both are *binary* classification tasks rather than three-way.

3.4 Experiments

Baselines We compare our calibrator against existing baselines as well as our own ablations.

MAXPROB simply uses the thresholded probability of the predicted class to assess whether the prediction is trustworthy.

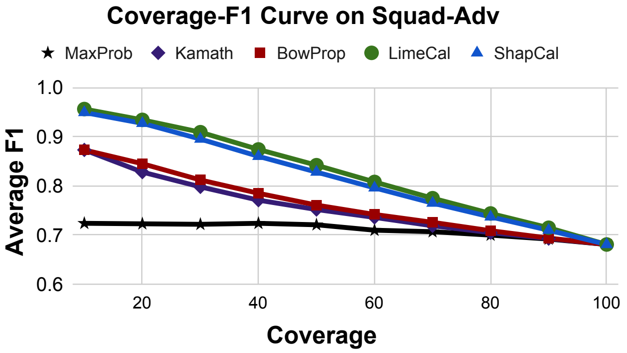


Figure 3.3: Coverage-F1 curves of different approaches on SQUAD-ADV. As more low-confidence questions are answered, the average F1 scores decrease. We use AUC to evaluate calibration performance.

KAMATH (Kamath et al., 2020) (for QA only) is a baseline initially proposed to distinguish out-of-distribution data points from in-domain data points in the selective QA setting (see Section 3.5), but it can also be applied in our settings. It trains a random forest classifier to learn whether a model's prediction is correct based on several heuristic features, including the probabilities of the top 5 predictions, the length of the context, and the length of the predicted answer. Since we are calibrating black box models, we do not use dropout-based features in Kamath et al. (2020).

CLSPROBCAL (for NLI only) uses more detailed information than MAXPROB: it uses the predicted probability for `Entailment`, `Contradiction`, and `Neutral` as the features for training a calibrator instead of only using the maximum probability.

BOWPROP adds a set of heuristic property features on top of the KAMATH method. These are the same as the features used by the full model *excluding the explanations*. We use this baseline to give a baseline for using general "shape" features on the inputs *not* paired with explanations.

Implementation of Our Method We refer our explanation-based calibration method using explanations produced by LIME and SHAP as **LIMECAL** and **SHAPCAL** respectively. We note that these methods also take advantages of the bag-of-word features in BOWPROP.

	SQuAD-Adv									
Approach	Acc	ΔBow	AUC	ΔBow	F1@25	ΔBow	F1@50	ΔBow	F1@75	ΔBow
MaxProb	62.6	–	70.9	–	72.4	–	72.1	–	70.4	–
Kamath	63.2	–	76.8	–	81.4	–	75.2	–	71.2	–
BowProp	63.6	0	77.4	0	82.9	0	76.1	0	71.7	0
LimeCal	**70.3**	6.7±1.6	**83.9**	6.4±1.4	**92.3**	9.4±2.3	**84.2**	8.1±1.6	**75.9**	4.2±1.0
ShapCal	69.3	5.6±1.8	82.9	5.5±1.3	91.2	8.2±2.2	82.8	6.7±1.4	75.0	3.3±0.9

	TriviaQA									
Approach	Acc	ΔBow	AUC	ΔBow	F1@25	ΔBow	F1@50	ΔBow	F1@75	ΔBow
MaxProb	67.0	–	76.7	–	82.1	–	76.3	–	71.0	–
Kamath	70.6	–	76.6	–	82.1	–	77.9	–	71.1	–
BowProp	71.2	0	77.6	0	84.2	0	79.1	0	71.6	0
LimeCal	**72.0**	0.8±0.4	**78.7**	1.1±0.2	**85.4**	1.2±0.8	**79.6**	0.5± 0.3	**72.3**	0.8±0.2
ShapCal	71.8	0.6±0.4	78.2	0.6±0.3	84.7	0.5±0.8	79.4	0.3± 0.4	72.3	0.8±0.3

	HotpotQA									
Approach	Acc	ΔBow	AUC	ΔBow	F1@25	ΔBow	F1@50	ΔBow	F1@75	ΔBow
MaxProb	63.1	–	75.7	–	79.7	–	75.9	–	72.2	–
Kamath	64.5	–	76.8	–	80.8	–	77.2	–	72.8	–
BowProp	64.7	0	76.6	0	80.3	0	76.9	0	72.4	0
LimeCal	**65.7**	1.0±0.4	**78.2**	1.6±0.4	**82.6**	2.2±0.8	**78.4**	1.5±0.6	**73.8**	1.4±0.3
ShapCal	65.3	0.7±0.4	77.8	1.2±0.3	82.0	1.6±0.7	78.0	1.0±0.5	73.5	1.1±0.4

Table 3.1: Main results on QA tasks. Our explanation-based methods (LimeCal and ShapCal) successfully calibrate a RoBERTa QA model trained on SQuAD when transferring to three new domains, and outperform a prior approach (Kamath) as well as our ablation using only heuristic labels (BowProp). In addition, we show the mean and standard deviation of the deltas w.r.t. BowProp across multiple random seeds in ΔBow.

For QA, the property space is the union of low-level `Segment` and `Segment` × `Pos-Tags`. For NLI, we use the union of `Segment` and `Segment` × `Pos-Tags` × `Overlapping Words` to label the tokens. Detailed numbers of features can be found in the Appendix.

3.4.1 Main Results: QA

Setup We train a RoBERTa (Liu et al., 2019) QA model on SQuAD as the base model, which achieves 85.5 exact match and 92.2 F1 score. For the experiments on HotpotQA and TriviaQA, we split the dev set, sample 500 examples for training, and leave the rest for testing. For experiments on SQUAD-ADV, we remove the unmodified data points in the ADD-SENT setting and also use 500 examples for training. For the experiments across all pairs, we randomly generate the splits, test the methods 20 times, and average the results to alleviate the influence of randomness.

Metrics In addition to *calibration accuracy* (**ACC**) that measures the accuracy of the calibrator, we also use the *area under coverage-F1 curve* (**AUC**) to evaluate the calibration performance for QA tasks in particular. The coverage-F1 curve (Figure 3.3) plots the average F1 score of the model achieved when the model only chooses to answer varying fractions (coverage) of the examples ranked by the calibrator-produced confidence. A better calibrator should assign higher scores to the questions that the models are sure of, thus resulting in higher area under the curve; note that AUC of 100 is impossible since the F1 is always bounded by the base model when every question is answered. We additionally report the average scores when answering the top 25%, 50%, and 75% questions, for a more intuitive comparison of the performance.

Results Table 3.1 summarizes the results for QA. First, we show that explanations are helpful for calibrating black box QA models out-of-domain. Our method using LIME substantially improves the calibration AUC compared to KAMATH by 7.1, 2.1 and 1.4 on SQUAD-ADV, TRIVIAQA, and HOTPOTQA, respectively. In particular, LIMECAL achieves an average F1 score of 92.3 at a coverage of 25% on SQUAD-ADV, close to the performance of the base model on original SQUAD examples. Our explanation-based approach is effective at identifying the examples that are robust with respect to the adversarial attacks.

Comparing LIMECAL against BOWPROP, we find that the explanations themselves do indeed help. On SQUAD-ADV and HOTPOTQA, BOWPROP performs on par with or only slightly better than KAMATH. These results show that connecting explanations with annotations is a path towards building better calibrators.

Finally, we compare the performance of our methods based on different explanation

	QNLI				MRPC			
Approach	Acc	ΔBow	AUC	ΔBow	Acc	ΔBow	AUC	ΔBow
MaxProb	50.5	–	41.2	–	57.0	–	50.0	–
ClsProbCal	56.7	–	59.5	–	71.5	–	77.9	–
BowProp	74.0	0	82.0	0	71.8	0	79.3	0
LimeCal	**75.0**	1.0±0.4	**82.6**	0.7±0.4	**73.6**	1.8±1.3	**81.0**	1.7±0.9
ShapCal	74.2	0.2±0.4	81.9	0.0±0.4	73.5	1.7±1.2	80.7	1.4±0.8

Table 3.2: Main results on NLI tasks. LimeCal moderately improves the performance of the base MNLI model on QNLI and MRPC, despite how different these tasks are from the base MNLI setting.

techniques. LimeCal slightly outperforms ShapCal in all three settings. As discussed in Section 3.2.1, Shap assigns high instance weights to those perturbations with few activated features. While such a choice of the kernel is effective in tasks involving tabular data (Lundberg and Lee, 2017), this might not be appropriate for the task of QA when such perturbations may not yield meaningful examples.

3.4.2 Main Results: NLI

Setup Our base NLI model is a RoBERTa classification model trained on MNLI, which achieves 87.7% accuracy on the development set. We collapse `contradiction` and `neutral` into `non-entailment` when evaluating on QNLI and MRPC. We continue using random forests as the calibrator model. We evaluate the generalization performance on the development sets of QNLI and MRPC. Similar to the settings in QA, we use 500 examples to train the calibrator and test on the rest for each of the 20 random trials.

Metrics Because QNLI and MRPC are binary classification tasks, predicting whether a model is correct (our calibration setting) is equivalent to the original prediction task. We can therefore measure calibrator performance with standard classification accuracy and AUC.

Results We show results on NLI tasks in Table 3.2. The base MNLI model utterly fails when transferring to QNLI and MRPC and achieves an accuracy of 49% and 57%, respectively, whereas the majority class is 50% (QNLI) and 65% (MRPC). With heuristic

Source \ Target		Sq-Adv	Trivia	Hotpot
Sq-Adv	Adapt	70.9	76.1	65.8
	Kamath		73.3	75.1
	BowProp		71.9	74.1
	LimeCal		72.9	71.4
Trivia	Adapt	64.2	76.7	**77.2**
	Kamath	70.5		76.7
	BowProp	67.1		75.0
	LimeCal	69.3		77.0
Hotpot	Adapt	56.6	74.0	75.7
	Kamath	70.6	77.0	
	BowProp	69.1	76.9	
	LimeCal	68.8	**77.9**	

Table 3.3: Area under Coverage-F1 curve for cross-domain calibration results. The numbers along the diagonal shows the MaxProb performance. A better performance than MaxProb suggests the calibrator is able to usefully generalize (colored cells).

annotations, BowProp is able to solve 74% of the QNLI instances and 72% of the MRPC instances. Our heuristic itself is strong for QNLI compared to MaxProb. LimeCal is still the best in both settings, moderately improving accuracy by 1% and 2% over BowProp using explanations. The results on NLI tasks suggest our method can still learn useful signals for indicating model reliability even if the underlying tasks are very different.

3.4.3 Analysis

Cross-Domain Generalization of Calibrators Our calibrators so far are trained on individual transfer settings. Is the knowledge of a calibrator learned on some initial domain transfer setting, e.g., SQuAD → TriviaQA, generalizable to another transfer setting, e.g. → HotpotQA? This would enable us to take our basic QA model and a calibrator and apply *that pair* of models in a new domain without doing any new training or adaptation. We explore this hypothesis on QA.[4]

For comparison, we also give the performance of a RoBERTa-model first finetuned

[4] We also tested the hypothesis on the NLI-paraphrase transfer, but did not see evidence of transferability there, possibly due to the fact that these tasks fundamentally differ.

	QA	100	300	500
Sq-Adv	MaxProb		70.9	
	Kamath	72.7	75.6	76.8
	BowProp	75.0	76.0	77.4
	LimeCal	**78.7**	**82.7**	**83.9**
Trivia	MaxProb		76.7	
	Kamath	74.8	76.2	76.6
	BowProp	76.1	77.4	77.6
	LimeCal	**77.2**	**78.2**	**78.7**
Hotpot	MaxProb		75.7	
	Kamath	75.2	76.5	76.8
	BowProp	74.9	76.3	76.6
	LimeCal	**76.5**	**77.7**	**78.2**
	NLI	100	300	500
QNLI	MaxProb		41.2	
	Kamath	56.4	58.1	59.5
	BowProp	79.0	81.5	82.0
	LimeCal	**79.1**	**81.8**	**82.8**
MRPC	MaxProb		50.0	
	Kamath	73.7	76.8	77.9
	BowProp	69.4	77.5	79.3
	LimeCal	**76.1**	**79.9**	**81.0**

Table 3.4: AUC scores of the calibrators trained with varying training data size. Explanation-based calibrators can still learn even with limited training resources, whereas Kamath and BowProp are not effective and underperform the MaxProb baseline on TriviaQA and HotpotQA.

on SQuAD and then finetuned on domain A (Adapt, Figure 3.2). Adapt requires access to the model architecture and is an unfair comparison for other approaches.

We show the results in Table 3.3. None of the approaches generalize between SQuAD-Adv and the other domains (either trained or tested on SQuAD-Adv), which is unsurprising given the synthetic and very specific nature of SQuAD-Adv.

Between TriviaQA and HotpotQA, both the LimeCal and Kamath calibrators trained on one domain can generalize to the other, even though BowProp is not effective. Furthermore, our LimeCal exhibits a stronger capability of generalization compared to Kamath. We then compare LimeCal against Adapt. Adapt does not always work

	SQuAD-Adv		TriviaQA		HotpotQA		QNLI	MRPC
Model Performance	Ex	F1	Ex	F1	Ex	F1	Acc	Acc
Base QA/NLI	62.1	68.0	53.2	62.1	50.7	66.3	50.5	57.2
Finetune RoBERTa	32.3	42.0	28.5	34.8	39.5	54.8	81.2	79.8
Adapt Base QA/NLI	77.3	84.3	56.2	64.0	54.3	70.8	80.7	79.1
InDomain QA/NLI	–	–	62.1	68.1	59.7	77.2	92.0	87.2
Calibration Results	Acc	AUC	Acc	AUC	Acc	AUC	Acc	Acc
Finetune RoBERTa + MaxProb	–	41.1	–	37.6	–	67.0	**81.2**	**79.8**
Adapt Base QA/NLI + MaxProb	–	**92.7**	–	77.6	–	**82.5**	80.7	79.1
LimeCal	69.3	82.9	72.0	**78.7**	65.7	78.2	74.9	73.6

Table 3.5: Model performance and calibration performance of LimeCal and glass box methods. On QA tasks, LimeCal is better than Finetuning RoBERTa and even outperforms Adapt Base QA/NLI on TriviaQA. LimeCal under-performs glass box methods on NLI due to its easy nature and the poor base-model performance.

well, which has also been discussed in Kamath et al. (2020); Talmor and Berant (2019). Adapt leads to a huge drop in terms of performance when being trained on HotpotQA and tested on TriviaQA, whereas LimeCal is the best in this setting. From TriviaQA to HotpotQA, Adapt works well, but Lime is almost as effective.

Overall, the calibrator trained with explanations as features exhibits successful generalizability across the two realistic QA tasks. We believe this can be attributed to the features used in the explanation-based calibrator. Although the task is different, the calibrator can rely on some common rules to decide the reliability of a prediction.

Impacts of Training Data Size Calibrating a model for a new domain becomes cumbersome if large amounts of annotated data are necessary. We experiment with varying the amount of training data the calibrator is exposed to, with results shown in Table 3.4. Our explanation-based calibrator is still the best in every setting with as few as 100 examples. With 100 examples, Kamath and BowProp perform worse than the MaxProb baseline on TriviaQA and HotpotQA, indicating that more data is needed to learn to use their features.

3.4.4 Comparison to Finetuned Models

Throughout this chapter, we have assumed a black box model that cannot be finetuned on a new domain. In this section, we compare calibration-based approaches with glass-box methods that require access to the model architectures and parameters. We evaluate two glass-box methods in two different settings (Figure 3.2): (1) finetuning a base RoBERTa model (Finetune RoBERTa), which needs access to the model's architecture but not parameters; and (2) finetuning a base QA/NLI model, which requires both model architectures as well as parameters. All these models are finetuned with 500 examples, the same as LimeCal. We also give the performance of a model trained with full in-domain training data for different tasks as references (InDomain QA/NLI).

We present the model performance (measured with Exact Match and F1 for QA and Acc for NLI) and calibration results in Table 3.5. Note that there are no calibrators for glass box methods, so we only report AUC scores for calibration performance.

On QA tasks, the limited training data is not sufficient for successfully finetuning a RoBERTa model. Consequently, Finetune RoBERTa does not achieve credible performance. Finetuning a base QA model greatly improves the performance, surpassing LimeCal on Squad-Adv and HotpotQA. However, we still find that on TriviaQA, LimeCal slightly outperforms Adapt. This is a surprising result, and shows that explanation-based calibrators can still be beneficial in some scenarios, even if we have full access to the model.

On NLI tasks that are substantially easier than QA, finetuning either a RoBERTa LM model or a base NLI model can reach an accuracy of roughly 80%. Our explanation-based approach largely lags glass-box methods, likely because the base NLI model utterly fails on QNLI (50.5% accuracy) and MRPC (55.0% accuracy) and does not grant much support for the two tasks. Nonetheless, the results on NLI still support our main hypothesis: explanations can be useful for calibration.

Known \ Unknown		Sq-Adv	Trivia	Hotpot
Sq-Adv	MaxProb	85.0	88.7	87.5
	Kamath	88.8	89.5	88.9
	BowProp	91.5	90.6	89.0
	LimeCal	**94.5**	**91.7**	**91.9**
Trivia	MaxProb	85.0	88.7	87.6
	Kamath	85.6	91.9	88.7
	BowProp	85.3	92.1	89.9
	LimeCal	**90.9**	**92.5**	**92.1**
Hotpot	MaxProb	85.0	88.7	87.6
	Kamath	86.1	91.4	89.4
	BowProp	85.1	91.8	91.6
	LimeCal	**91.7**	**92.3**	**92.5**

Table 3.6: Area under Coverage-F1 curve in the Selective QA setting. Our explanation-based approach is also strong in this setting, substantially outperforming existing baseline and our own ablation.

3.5 Selective QA Setting

Our results so far have shown that a calibrator can use explanations to help make binary judgments of correctness for a model running in a new domain. We now test our model on the selective QA setting from Kamath et al. (2020) (Figure 3.2). This experiment allows us to more directly compare with prior work and see performance in a setting where in-domain (ID) and out-of-domain (OOD) examples are mixed together.

Given a QA model trained on source domain data, the goal of selective QA is to train a calibrator on a mixture of ID source data and *known* OOD data, and test the calibrator to work well on a mixture of in-domain and an *unknown* OOD data.

We follow the similar experimental setup as in Kamath et al. (2020). The detailed setting is included in the supplementary material.

Results As shown in Table 3.6, similar to the main QA results. Our explanation-based approach, LimeCal, is consistently the best among all settings. We point out our approach outperforms Kamath especially in settings that involve SQuAD-Adv as known or unknown OOD distribution. This can be attributed the similarity between SQuAD and SQuAD-Adv which can not be well distinguished with features used in KAMATH (`Context Length`, `Answer Length`, and etc.). The strong performance of our explanation-based approach in the selective QA setting further verifies our assumption: explanation can be useful and effective for calibrating black box models.

3.6 Related Work

Our approach is inspired by recent work on the *simulation* test (Doshi-Velez and Kim, 2017), i.e., whether humans can simulate a model's prediction on an input example based on the explanations. Simulation tests have been carried out in various tasks (Ribeiro et al., 2018; Nguyen, 2018; Chandrasekaran et al., 2018; Hase and Bansal, 2020) and give positive results in some tasks (Hase and Bansal, 2020). Our approach tries to mimic the process that humans would use to judge a model's prediction by combining heuristics with attributions instead of having humans actually do the task.

Using "meta-features" to judge a model also appears in literature on system combination for tasks like machine translation (Bojar et al., 2017), question answering (Kamath et al., 2020; Zhang et al., 2021), constituency parsing (Charniak and Johnson, 2005; Fossum and Knight, 2009) and semantic parsing (Yin and Neubig, 2019). The work of Rajani and Mooney (2018) in VQA is most relevant to ours; they also use heuristic features, but we further conjoin heuristic with model attributions. Our meta-feature set is derived from the presence of certain properties, which is similar to the "concepts" used in concept-based explanations (Ghorbani et al., 2019; Mu and Andreas, 2020), but we focus on using them for estimating model performance rather than explaining a prediction.

Our work addresses the problem of calibration (Guo et al., 2017; Desai and Durrett, 2020), which is frequently framed in terms of models' output probabilities. Past work has attempted to tackle this problem using temperature scaling (Guo et al., 2017) or label smoothing (Pereyra et al., 2017), which adjust confidence scores for all predictions. In contrast, we approach this issue by applying a classifier leveraging instance-specific explanations. Past work on generalizing to out-of-domain distribution in NLP largely focuses on using unlabeled data from the target domain and requires finetuning a model (Ma et al., 2019; Ramponi and Plank, 2020; Guo et al., 2020), whereas we improve OOD performance of strictly black-box models.

3.7 Discussion & Conclusion

Limitations Despite showing promising results in improving model generalization performance, our attribution-based approach does suffer from intensive computation cost. Using either LIME or SHAP to generate attributions requires running inference a fair number of perturbations when the input size is large (see Appendix for details), which limits our method's applicability. But this doesn't undermine the main contribution of the method in this chapter, answering the question in the title, and our approach is still applicable as-is in the scenarios where we pay for access to the model but not per query.

The evaluation of calibration in this chapter primary focuses on *relative* confidences: a system is considered to be better calibrated if it assigns higher confidence scores to correct predictions than to incorrect ones. We do not assess the absolute confidence scores using expected calibration error (ECE), the standard metric used in past work on calibration (Guo et al., 2017; Zhang et al., 2021). Part of the reason is that, for the QA datasets studied in our work, a prediction can be partially correct (with an F1 score greater than 0.0 and less than 1.0), whereas ECE considers only the binary correctness (either 0 or 1) of predictions. We leave improving the absolute calibration of confidence scores in addition to the relative ranking as future work.

Conclusion We have explored whether model attributions can be useful for calibrating black box models. The answer is *yes*. By connecting attributions with human heuristics, we improve model generalization performance on new domains and tasks. Besides, it exhibits promising generalization performance in some settings (cross-domain generalization and Selective QA).

Chapter 4: Calibrating In-Context Learning Using Explanations[1]

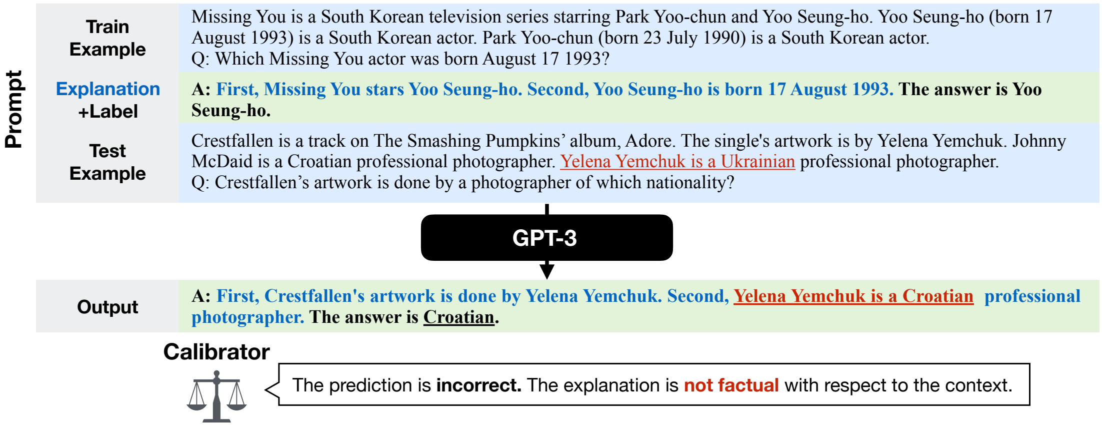
4.1 Introduction

In Chapter 2 and 3, we show the utility of attributions for improving BERT-based LMs. This chapter focuses instead on large language models (LLMs), and shows that the verification framework proposed in Chapter 3 can also be applied for improving LLMs with free-text explanations.

A unique emerging capability of LLMs that have not been possessed by earlier BERT-based models is in-context learning: LLMs are able to learn NLP tasks from just a few training examples "in context," without updating the model's parameters (Brown et al., 2020). However, this learning process is still poorly understood: models are biased by the order of in-context examples (Zhao et al., 2021) and may not leverage the instructions or even the labels of the examples in the ways one expects (Min et al., 2022; Webson and Pavlick, 2022). Existing tools for interpreting model predictions have high computational cost (Ribeiro et al., 2016) or require access to gradients (Simonyan et al., 2014; Sundararajan et al., 2017), making them unsuitable for investigating in-context learning or explaining the predictions of prompted models.

One appealing way to gain more insight into predictions obtained through in-context learning is to let the language model "explain itself" (Nye et al., 2021; Wei et al., 2022c; Chowdhery et al., 2022; Marasović et al., 2022; Lampinen et al., 2022). In addition to input-label training pairs in context, one can prompt the language model with an explanation for each pair and trigger the model to generate an explanation for its prediction (Figure 4.1). Prompting with explanations introduces much richer information compared to using la-

[1] An early version of this chapter has been published in Ye and Durrett (2022b). Xi Ye is the first author of Ye and Durrett (2022b), where he developed the research idea with other author(s), implemented the framework, designed and performed the experiments and analysis, and wrote the paper.



Missing You is a South Korean television series starring Park Yoo-chun and Yoo Seung-ho. Yoo Seung-ho (born 17 August 1993) is a South Korean actor. Park Yoo-chun (born 23 July 1990) is a South Korean actor.
Q: Which Missing You actor was born August 17 1993?

A: First, Missing You stars Yoo Seung-ho. Second, Yoo Seung-ho is born 17 August 1993. The answer is Yoo Seung-ho.

Crestfallen is a track on The Smashing Pumpkins' album, Adore. The single's artwork is by Yelena Yemchuk. Johnny McDaid is a Croatian professional photographer. Yelena Yemchuk is a Ukrainian professional photographer.
Q: Crestfallen's artwork is done by a photographer of which nationality?

A: First, Crestfallen's artwork is done by Yelena Yemchuk. Second, Yelena Yemchuk is a Croatian professional photographer. The answer is Croatian.

The prediction is **incorrect.** The explanation is **not factual** with respect to the context.

Figure 4.1: Prompting GPT-3 with explanations. By including explanations in the in-context examples, we can cause GPT-3 to generate an explanation for the test example as well. In this case, the generated explanation is nonfactual, despite the simple reasoning involved here. However, we show this nonfactuality actually provides a signal that can help calibrate the model.

bels alone, which might guide the inference process and allow the model to learn more information from the examples.

In this chapter, we investigate the nature of the explanations that LLMs generate and whether they can improve few-shot in-context learning for textual reasoning tasks, specifically QA and NLI. Recent prior work that finds success with this approach largely targets symbolic reasoning tasks with a very different structure, such as math word problem solving (Nye et al., 2021; Wei et al., 2022c). We experiment on three different datasets spanning QA and NLI with four LLMs: OPT, GPT-3 (davinci), InstructGPT (text-davinci-001), and text-davinci-002. The results suggest that explanations only substantially improve accuracy for text-davinci-002, but give a smaller improvement or even hurt the performance with the other LLMs.

Surprisingly, we find that the explanations generated by LLMs can be **unreliable**, even for a very simple synthetic dataset. We evaluate the explanations along two axes: *factuality*, whether the explanation is correctly grounded in the input, and *consistency*, whether the explanation entails the final prediction. LLMs tend to generate consistent explanations that account for the predictions, but the explanations may not be factual, as as shown in Figure 4.1. Furthermore, our analysis suggests an unreliable explanation more likely indicates a wrong prediction compared to a reliable explanation.

Despite LLMs' failures here, we can still benefit from model-generated explanations by using them for calibration. If we are able to automatically assess the reliability of an explanation, we can allow an LLM to return a null answer when its explanation is unreliable, since the prediction in this case is less likely to be correct. Unfortunately, there is no automated way to perfectly assess the reliability, but we can extract features that approximately reflect it. We use these features to calibrate InstructGPT's[2] predictions, and successfully improve the in-context learning performance across all the datasets.

In summary, our main findings are: (1) Simply plugging explanations into the prompt does not always substantially boost the in-context learning performance for textual reasoning. (2) LLMs generate explanations consistent with their predictions, but these explanations might not be factually grounded in the inputs. (3) The factuality of an explanation can serve as an indicator for the correctness of the corresponding prediction. (4) Using features that can approximate the factuality of explanations, we successfully use explanations to improve the in-context learning performance across all tasks.

4.2 Does Prompting with Explanations Improve In-Context Learning?

In this chapter, we specifically focus on tasks involving reasoning over natural language. These are tasks where explanations have been traditionally studied (Camburu et al., 2018; Rajani et al., 2019), but which are more complex than tasks like sentiment analysis which are well explained by extractive rationales (Zaidan et al., 2007; DeYoung et al., 2020). We experiment on two tasks, reading comprehension question answering (QA) and natural language inference (NLI), on three English-language datasets. For each dataset,

[2] Throughout this chapter, we primarily test on InstructGPT for two reasons. First, it was the most capable model available at the time we were conducting the majority of our experiments. Second, it still has significant room to improve on the datasets we explore in this chapter. This setting is a representative testbed for the situation where an LLM-based system does not yet give satisfactory performance on a target task, causing the system designer to turn to explanations in prompts to improve things.

SYNTH	**Context:**	Christopher agrees with Kevin. Tiffany agrees with Matthew. Mary hangs out with Danielle. James hangs out with Thomas. Kevin is a student. Matthew is a plumber. Danielle is a student. Thomas is a plumber.
	Question:	Who hangs out with a student?
	Answer:	Mary **Explanation:** Danielle is a student and Mary hangs out with Danielle.
E-SNLI	**Premise:**	A toddler in a green jersey is being followed by a wheelchair bound woman in a red sweater past a wooden bench.
	Hypothesis:	A toddler is walking near his wheelchair bound grandmother.
	Label:	Neither **Explanation:** the woman may not be his grandmother.

Figure 4.2: A SYNTH example and an E-SNLI example. See Figure 4.3 for HOTPOTQA examples.

we create a test set with 250 examples.

4.2.1 Datasets

Synthetic Multi-hop QA (SYNTH) In order to have a controlled setting where we can easily understand whether explanations are factual and consistent with the answer, we create a synthetic multi-hop QA dataset. Shown in Figure 4.2, each example in this dataset asks a bridge question (using the terminology of (Yang et al., 2018)) over a context consisting of supporting facts paired with controlled distractors. This dataset is carefully designed to avoid spurious correlations, giving us full understanding over the correct reasoning process and the explanation for every example, which naturally consists of the two supporting sentences. [3]

Adversarial HotpotQA (HOTPOTQA) We also test on the English-language Adversarial HotpotQA dataset (Yang et al., 2018; Jiang and Bansal, 2019). We use the adversarially augmented version since InstructGPT achieves high performance on the distractor setting of the original dataset. We make a challenging set of examples by balancing sets of questions on which InstructGPT makes correct and incorrect predictions. The context of each question includes two ground truth supporting paragraphs and two adversarial paragraphs.

[3] This dataset is inspired by task 15 of the bAbI dataset (Weston et al., 2016). In our preliminary experiments with some of the other bAbI tasks, we found poor performance from InstructGPT similar to our results on SYNTH, both with and without explanations.

For HOTPOTQA, we manually annotated explanations for the training examples. Figure 4.1 shows an example of such an explanation, highlighted in orange. We could use the supporting sentences as the explanations, but we found they are usually too verbose and not sufficient, e.g., with anaphors that resolve outside of the supporting sentences. Therefore, we manually annotate a set of explanations which clearly describe the reasoning path for each question.

E-SNLI E-SNLI (Camburu et al., 2018) is an English-language classification dataset commonly used to study explanations, released under the MIT license. Shown in Figure 4.2, each example consists of a premise and a hypothesis, and the task is to classify the hypothesis as entailed by, contradicted by, or neutral with respect to the premise. As a notable contrast to the other datasets, the explanations here are more *abstract* natural language written by human annotators, as opposed to mostly constructed from extracted snippets of context.

4.2.2 Baselines

We study the effectiveness of plugging in explanations by comparing the in-context learning performance of prompting with or without explanations. Prompting without explanations resembles the standard few-shot in-context learning approach (**Few-Shot**). To incorporate explanations into the prompt, we consider the following two most commonly used paradigms:

Explain-then-Predict (E-P) prepends an explanation before the label (Figure 4.1). The language model is expected to generate an explanation first followed by the prediction. The prompting style of past work involving computational traces can be categorized into this paradigm, including Nye et al. (2021) and Wei et al. (2022c). This approach is also called a pipeline model in other literature on training models using explanations (Jacovi and Goldberg, 2021; Wiegreffe et al., 2021).

Predict-then-Explain (P-E) generates the explanation after the prediction. Unlike E-P, the predicted explanation does not influence the predicted label, since we use greedy inference and the explanation comes afterwards. However, the explanations in the prompt still impact the predictions.

4.2.3 Setup

For few-shot learning, we use roughly the maximum allowed shots in the prompt that can fit the length limit of OPT (Zhang et al., 2022a) and GPT-3 (Brown et al., 2020), which is 16 for SYNTH, 6 for HOTPOTQA, and 32 for E-SNLI, respectively.[4] We experiment with four LLMs, including OPT (175B), GPT-3 (davinci), InstructGPT (text-davinci-001), and text-davinci-002. OPT and GPT-3 are trained using the standard causal language modeling objective, whereas InstructGPT and text-davinci-002 are trained with special instruction data and human annotations. We generate outputs with greedy decoding (temperature set to be 0). Our prompt formats follow those in Brown et al. (2020). The explanations are inserted before/after the prediction with conjunction words like *because*. Because the results of in-context learning vary with the examples presented in the input prompt, for each dataset, we randomly sample multiple groups of training shots, and report the mean and standard deviation of the results (subscript). We use 5 groups for InstructGPT, the primary LM we are using throughout this chapter, and 3 groups for the rest.

4.2.4 Results

As shown in Table 4.1, OPT, GPT-3, and InstructGPT show small to moderate improvements from using explanations for textual reasoning tasks. On the two QA tasks, SYNTH and HOTPOTQA, E-P improves the performance of InstructGPT, the best among these three LMs, from 54.8 to 58.5 and 56.8 to 59.4, respectively.[5] On E-SNLI, P-E outperforms FEW-SHOT by 2.6, whereas E-P substantially lags FEW-SHOT. Comparing E-P against P-E on SYNTH and E-SNLI, E-P typically degrades performance (except

[4] This contrasts with recent work like Zhao et al. (2021) that focuses on improving performance in the 1-4-shot setting; by using more data we achieve much stronger results on our tasks.

[5] For SYNTH, we also tried using an alternative style of explanations (reversing the order of the two sentences in the explanations), which leads to mild performance degradation.

Table 4.1: Results of prompting with explanations on four large language models. Using explanations leads to small to moderate improves performance on OPT, GPT-3, and InstructGPT, and has more prominent effects on text-davinci-002.

		Synth	HotpotQA	e-SNLI
OPT (175B)	Few-Shot	$\mathbf{40.5}_{2.8}$	$49.7_{2.6}$	$\mathbf{44.0}_{3.8}$
	E-P	$29.6_{0.5}$	$\mathbf{52.6}_{6.5}$	$39.3_{7.8}$
	P-E	$40.2_{2.6}$	$43.3_{4.5}$	$43.4_{1.6}$
GPT-3	Few-Shot	$49.5_{0.6}$	$49.1_{6.2}$	$43.3_{5.7}$
	E-P	$47.1_{2.8}$	$\mathbf{54.1}_{4.1}$	$40.4_{4.5}$
	P-E	$\mathbf{51.3}_{1.8}$	$48.7_{4.6}$	$\mathbf{48.7}_{2.4}$
InstructGPT	Few-Shot	$54.8_{3.1}$	$53.2_{2.3}$	$56.8_{2.0}$
	E-P	$\mathbf{58.5}_{2.1}$	$\mathbf{58.2}_{4.1}$	$41.8_{2.5}$
	P-E	$53.6_{1.0}$	$51.5_{2.4}$	$\mathbf{59.4}_{1.0}$
text-davinci-002	Few-Shot	$72.0_{1.4}$	$77.7_{3.2}$	$69.1_{2.0}$
	E-P	$\mathbf{86.9}_{3.8}$	$\mathbf{82.4}_{5.1}$	$\mathbf{75.6}_{7.6}$
	P-E	$81.1_{2.8}$	$77.2_{4.8}$	$69.4_{5.0}$

on Synth for InstructGPT) and P-E is inconsistent across the different models, whereas E-P consistently leads to performance improvements on HotpotQA. There is no single winner between the two paradigms of using explanations; choosing the most effective way is task-specific. Overall, vanilla LLMs (OPT and GPT-3) see limited benefit from producing explanations, and even the Instruct-series InstructGPT does not see substantial improvements.

The only exception is text-davinci-002. text-davinci-002 greatly benefits from explanations in the prompt across all three tasks, and E-P is consistently more effective than P-E. However, it is unclear what contributes to this difference. As far as we are aware, the differences between text-davinci-002 and InstructGPT are not described in any publication or blog post.[6] Comparing GPT-3 and InstructGPT, we see the move to Instruct series models is *not* sufficient to explain the difference. Given the lack of transparency with this model,

[6] One publicly-described difference is the addition of editing and insertion, discussed at `https://openai.com/blog/gpt-3-edit-insert/`, but this does not explain the performance differences we observe.

Nonfactual	Pedro Rubens! The individual Chapters were published into 64 ”tankōbon” by Kodansha. Yōko Shōji (born 4 June 1950, in Mobara, Chiba) is a Japanese manga artist. She is best known for writing ”Seito Shokun! Mulder Scully! The individual Chapters were published into 14 ”tankōbon” by Kodansha. Seito Shokun! The individual Chapters were published into 24 ”tankōbon” by Kodansha between. Q: How many Chapters does Yōko Shōji’s most famous manga have? A: First, Yōko Shōji’s most famous manga is ”Seito Shokun!”. Second, ”Seito Shokun!” has 64 Chapters. The answer is 64.
Inconsistent	Tim Minchin (December 29, 1808 July 31, 1875) was the President of the United States. Andrew Johnson (December 29, 1808 July 31, 1875) was the President of the United States. George Andrew Atzerodt (June 12, 1835 – July 7, 1865) was a conspirator, with John Wilkes Booth. Jesse Andrew Williams (June 12, 1835 – July 7, 1865) was a conspirator, with John Wilkes Booth. Q: Who was older, George Atzerodt or Andrew Johnson? A: First, George Atzerodt was born on June 12, 1835. Second, Andrew Johnson was born on December 29, 1808. The answer is George Atzerodt.

Figure 4.3: Explanations generated for HOTPOTQA. InstructGPT may generate nonfactual explanations containing hallucination (red) or inconsistent explanations contradicting the answer (red).

we hesitate to make scientific claims about the results it yields.

Our results do not suggest immediate strong improvements from incorporating explanations across all LLMs, even for our synthetic dataset, contradicting recent prior work. This can be attributed to the difference between the tasks we study. The tasks that receive significant benefits from using explanations in Nye et al. (2021) and Wei et al. (2022c) are all program-like (e.g., integer addition and program execution), whereas the tasks in this chapter emphasize textual reasoning grounded in provided inputs. In fact, in Wei et al. (2022c) and Chowdhery et al. (2022), explanations only show mild benefit on open-domain QA tasks like StrategyQA (Geva et al., 2021b) that are closer to our setting.

4.3 Can LLMs Generate Factual and Consistent Explanations?

Prompting LLMs with explanations and having models generate them may not guarantee higher performance on our tasks. But what about the quality of the model-generated explanations themselves? We assess the reliability of the explanations for the three datasets, measured in terms of two aspects.

Factuality refers to whether a generated explanation is faithfully grounded in the corresponding input context (context for QA and premise/hypothesis pair for NLI). A factual

Table 4.2: Left: factuality (Fac) and consistency (Con) of the generated explanations. Right: the % of the examples whose explanation factuality/consistency is congruent with the prediction accuracy. In general, LLMs tend to generate consistent but less likely factual explanations.

		Acc	**Fac**	**Con**	**Acc=Fac**	**Acc=Con**
	reliability of explanations generated by InstructGPT					
InstructGPT	Synth (E-P)	58.4	72.8	64.8	66.5	68.8
	Synth (P-E)	54.8	51.6	95.2	**89.6**	57.2
	AdvHP (E-P)	62.0	79.6	91.2	**80.0**	68.4
	AdvHP (P-E)	54.0	69.2	82.0	**77.6**	67.2
	e-SNLI (P-E)	62.0	–	98.8	–	62.0
	reliability of explanations generated by other LLMs on Synth					
OPT (175B)	Synth (E-P)	30.0	77.2	47.2	45.6	58.8
	Synth (P-E)	39.6	64.0	81.2	**69.2**	49.6
GPT-3	Synth (E-P)	46.8	59.2	64.8	**66.8**	61.2
	Synth (P-E)	52.4	52.4	83.2	**78.4**	58.0
text-davinci-002	Synth (E-P)	86.0	91.6	85.2	**91.2**	84.8
	Synth (P-E)	81.6	83.2	96.4	**95.8**	82.8

explanation should not contain hallucinations that contradict the context. See Figure 4.3 for a nonfactual explanation.

Consistency measures if the explanation entails the prediction. Our concept of consistency resembles plausibility as described in Jacovi and Goldberg (2021), in that we assess whether the prediction follows from the explanation **as perceived by a human**. See Figure 4.3 for an inconsistent explanation.

For Synth, we use rules to automatically judge whether an explanation is factual and consistent on all four LLMs. For HotpotQA and e-SNLI, the authors manually inspected the explanations generated by InstructGPT and annotated them for these two characteristics. Note for each setting, the results are based on the explanations and predictions obtained with a single set of training shots. We only show the results of P-E on e-SNLI, as E-P is substantially worse here.

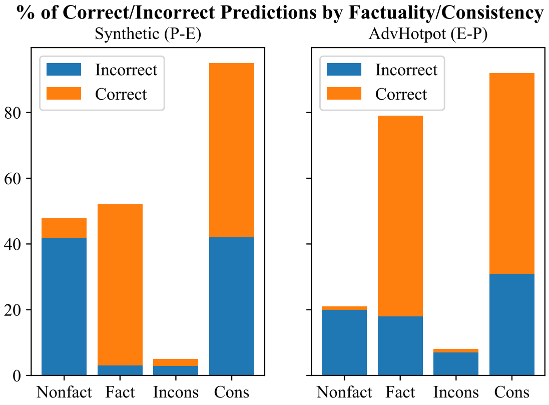


Figure 4.4: Explanations are more likely to be nonfactual than to be inconsistent, and a nonfactual explanation usually indicates an incorrect prediction.

Results We summarize the results in Table 4.2. We only report consistency on E-SNLI, as the explanations for E-SNLI often require some external commonsense knowledge which cannot be easily grounded in the inputs or judged as true or false. The results suggest a disconnect between the model predictions and the "reasoning" in explanations. On InstructGPT, though using explanations improves its performance across three tasks, the generated explanations are *unreliable* (upper section), even for the straightforward synthetic setting. Comparing the factuality of explanations for SYNTH generated by GPT-3, InstructGPT, and text-davinci-002, we see that instruction tuning improves the factuality, but even the most powerful text-davinci-002 still fails to generate explanations that are perfectly grounded in the input context. Overall, LLMs tend to generate consistent explanations (more than 80% for all three datasets with the right prompt structure), but the explanations are less likely to be factual, which is concerning as they can deceive a user of the system into believing the model's answer.

4.3.1 Reliability of Explanations and Prediction Accuracy

LLMs may hallucinate problematic explanations, but this could actually be advantageous if it gives us a way of spotting when the model's "reasoning" has failed. We investigate the connection between the reliability of an explanation and the accuracy of a prediction and ask whether a reliable explanation indicates an accurate prediction. (This resembles the linguistic calibration of Mielke et al. (2020), but using a different signal for calibration.)

As shown in Table 4.2 (right), accuracy and factuality/consistency are typically correlated, especially factuality. By knowing whether an explanation is factual, we can guess the model's accuracy a high fraction of the time (Accuracy = Factuality). A nonfactual explanation very likely means an incorrect prediction on the SYNTH dataset across all four LLMs. On HOTPOTQA, factuality and InstructGPT's prediction correspond 80.0% of the time, substantially surpassing the prediction accuracy itself. We show fractions of correct and incorrect predictions when the explanations are factual/nonfactual and consistent/inconsistent in Figure 4.4 for two of our settings. Factual explanations are much more likely paired with correct predictions compared to nonfactual explanations. Consistency is also connected to accuracy but is an inferior indicator compared to factuality in general (Table 4.2).

4.4 Calibrating In-Context Learning using Explanations

From Section 4.3.1, we see that a human oracle assessment of the factuality of an explanation could be of substantial use for calibrating the corresponding prediction. Can we automate this process?

We first show how to achieve this goal on the perfectly controlled SYNTH dataset (Section 4.4.1). On our other two datasets, we use surface lexical matching to approximate semantic matching and give real-valued scores approximately reflecting factuality. Following past work on supervised calibration (Kamath et al., 2020; Chen et al., 2021a), we can learn a calibrator that tunes the probabilities of a prediction based on the score of its explanation (Section 4.4.2). We show such a calibrator can be trained with a handful of examples beyond those used for in-context learning and successfully improve the in-context learning performance on realistic datasets.[7] We note that, as mentioned before, the experiments in

[7] This procedure does require extra data. However, it provides a natural avenue for using a small number of additional examples that otherwise would be *impossible* to incorporate into this procedure, when the size of the context actually limits the amount of data for in-context learning.

this section are conducted on InstructGPT.

4.4.1 Motivating Example: Improving SYNTH Dataset

We first show how post-hoc calibration functions in the controlled SYNTH setting, where we can simply check the factuality of an explanation. Since the generated explanation always follows the format "`B is [profession] and A [verb] B.`" (example in Figure 4.2), we can split the explanation into two sentences. The explanation is factual if and only if each of the two sentences exactly matches one of the sentences in the context.

We use the assessment to improve the performance of P-E for SYNTH, where a nonfactual explanation typically indicates an incorrect prediction. This gives us a way to reject presumably incorrect answers. Specifically, we iterate through the top 5 candidate answers (restricted by the API) given by InstructGPT and reject any answer-explanation pair if the explanation is nonfactual until we find a factual one. This procedure dramatically improves the accuracy from 52.4% to 74.8%. Note that this SYNTH dataset is a challenging task given its lack of reasoning shortcuts: for reference, neither ROBERTA (Liu et al., 2019) nor DEBERTA (He et al., 2021) finetuned with 16 examples can achieve an accuracy surpassing 50%. With the help of the explanations and the checking procedure, we can use InstructGPT to achieve strong results using few-shot learning.

4.4.2 Learning-based Calibration Framework

Framework We now introduce the framework that can leverage the factuality assessment of an explanation to calibrate a prediction. Let $\boldsymbol{p}$ be the vector of predicted probabilities associated with each class label in NLI (or the probability score of predicted answer in QA). Let v be a scalar value extracted from the explanation to describe the factuality. Then, we can adjust the probabilities accordingly using a linear model: $\hat{\boldsymbol{p}} = \mathrm{softmax}(W[\boldsymbol{p}; v] + b)$, where $\hat{\boldsymbol{p}}$ is the tuned probabilities.

Our calibration framework is extended from classical calibration methods (Platt, 1999; Guo et al., 2017; Zhao et al., 2021), which apply an affine transformation on the probabilities alone: $\hat{\boldsymbol{p}} = \mathrm{softmax}(W\boldsymbol{p} + b)$. In contrast, we use an additional factor v in calibration to incorporate the factuality assessment of the explanation.

There are a small number of parameters (W and b) that need to be trained in such a calibration framework. We will rely on a few more examples in addition to the shots we use in the prompt to train the calibrator. Specifically, we use the prompt examples to generate the predictions and explanations for these extra examples, and extract predicted probabilities, factors, and target probabilities triples to construct training data points used to train the calibrator. Note this procedure requires **no** explanation annotations for the extra examples.

Approximating Factuality We approximate the factuality using lexical overlap between the explanations and the inputs, which we found to work fairly well for our tasks.

HOTPOTQA: We use an explanation consisting of two sentences (examples in Figure 4.3) as an illustration. Let $\mathcal{E} = (E^{(1)}, E^{(2)})$ be the generated explanation, where $E^{(1)}$ and $E^{(2)}$ are the two sentences, and the $E^{(i)} = (e_1, e_2, \cdots)$ contain tokens $e_1, e_2, \cdots$. Similarly, let $\mathcal{P} = (P^{(1)}, P^{(2)}, P^{(3)}, P^{(4)})$ be the context paragraphs, and $P^{(i)} = (p_1, p_2, \cdots)$ be the tokens. The factuality estimation of one explanation sentence $E^{(i)}$ is defined as: $\mathcal{V}(E^{(i)}) = \max_{P \in \mathcal{P}} \frac{|E^{(i)} \cap P|}{|E^{(i)}|}$.

Intuitively, the factuality score for a sentence E is defined as the maximum number of overlapping tokens over all paragraphs P, normalized by the number of tokens in E. We then define the factuality score for the whole explanation as $\mathcal{V}(\mathcal{E}) = \min_{E \in \mathcal{E}} \mathcal{V}(E)$, as it requires all sentences to be factual in order to make the entire explanation factual.[8]

E-SNLI: The explanations of E-SNLI do not really involve a concept of factuality. Nevertheless, we use an analogous score following the same principle by viewing the premise as the context. Let $E = (e_1, e_2, \cdots)$ be the explanation and $P = (p_1, p_2, \cdots)$ be

[8] Alternatively, one might use a fine-tuned NLI model as a proxy (Chen et al., 2021a). However, our focus is on the pure black-box setting, and we avoid models that require substantial amounts of data to make work.

Table 4.3: Accuracy ($\text{mean}_{\text{std dev}}$) of various methods on E-SNLI under different data conditions. **L** denotes number of labels (as well as the total number of examples); **E** denotes the number of explanations. Calibrating using explanations successfully improves the performance of in-context learning.

w/o Explanation	**32L**	**64L**	**96L**	**128L**
RoBERTa	$40.1_{4.7}$	$43.0_{5.1}$	$49.0_{5.2}$	$54.9_{4.8}$
FEW-SHOT	$56.8_{2.0}$	–	–	–
FEW-SHOT(NN)	–	–	–	$58.9_{1.0}$
FEW-SHOT+PROBCAL	$61.9_{3.8}$	$62.4_{2.6}$	$63.2_{2.9}$	$63.9_{1.2}$
w/ Explanation	**32L+32E**	**64L+32E**	**96L+32E**	**128L+32E**
P-E	$59.4_{2.0}$	–	–	–
P-E+PROBCAL	$\mathbf{64.4_{1.8}}$	$65.4_{1.2}$	$65.4_{1.6}$	$65.4_{1.9}$
P-E+EXPLCAL	$64.2_{2.6}$	$\mathbf{65.8_{1.3}}$	$\mathbf{67.6_{1.6}}$	$\mathbf{68.5_{1.2}}$
P-E+ZHANG	$63.0_{3.2}$	$65.2_{2.2}$	$65.4_{1.5}$	$65.9_{2.5}$

the premise. We simply score the explanation by $\mathcal{V}(E) = \frac{|E|\cap|P|}{|E|}$. The more an explanation overlaps with the premise, the more factual we judge it to be.

4.4.3 Calibrating E-SNLI

Setup For E-SNLI, we use calibration methods to postprocess the final probabilities. Unlike classical temperature scaling (Platt, 1999), note that the methods we use here can actually change the prediction; we will therefore evaluate on *accuracy* of the calibrated model.

We study the effectiveness of our explanation-based calibrator under different training data sizes varying from 32 to 128. Recall that we only require explanation annotations for 32 data points, and only need the labels for the rest to train the calibrator. For E-SNLI, we calibrate P-E, which is shown to be more effective than E-P in this setting (Section 4.2.4).

Baselines We provide the performance of fine-tuned ROBERTA (Liu et al., 2019) model as a reference, finding this to work better than DeBERTa (He et al., 2021). To isolate the effectiveness of using explanations for calibration, we introduce three additional baselines using non-explanation-based calibrators. We apply the probability-based calibrator as described in Section 4.4.2 on the results obtained on few-shot learning (FEW-SHOT+PROBCAL) and predict-then-explain pipeline (P-E+PROBCAL). We note that the parameters of these calibrators are trained using the additional data points, as opposed to being heuristically determined as in Zhao et al. (2021). Furthermore, we experiment with a recently proposed supervised calibrator from Zhang et al. (2021), which uses the CLS representations from an additional language model as features in the calibrator. The probabilities are tuned using $\hat{\boldsymbol{p}} = \mathrm{softmax}(W[\boldsymbol{p}; \boldsymbol{h}] + b)$, where $\boldsymbol{h}$ is the CLS representation. Since we do not have access to the embeddings obtained by GPT-3, we use ROBERTA to extract the vectors instead. We use such a calibrator on top of our best-performing base model, P-E, resulting P-E+ ZHANG ET AL. (2021).

Limited by the maximum prompt length, in-context learning is not able to take as input the additional data used for training the calibrator. For a fair comparison, we can allow the in-context model to use this data by varying the prompts across test examples, dynamically choosing the prompt examples to maximize performance. Choosing closer data points for prompting is a common and effective way of scaling up the training data size for in-context learning (Shin et al., 2021; Liu et al., 2022). Following Liu et al. (2022), we test the performance of choosing nearest neighbors for the prompt based on CLS embedding produced by a ROBERTA model (Liu et al., 2019), referred as FEW-SHOT(NN). It is worth clarifying that the FEW-SHOT and FEW-SHOT+PROBCAL approaches use the same set of 32 training shots in the prompt for every test example, whereas the shot sets vary from example to example in FEW-SHOT(NN).

Results We show the results in Table 4.3. We use 5 different groups of training examples and report the mean and standard deviation across the groups. For FEW-SHOT(NN), we only report the results obtained using 128 examples; results using a smaller number of examples will be worse than this.

Under 128 training examples, applying a trained calibrator on top of prompting with explanation (i.e., P-E+EXPLCAL) achieves the best accuracy of 68.5%, which is 12%

higher than the performance of the vanilla uncalibrated few-shot in-context learning (FEW-SHOT). P-E+EXPLCAL also outperforms FEW-SHOT+PROBCAL and P-E+PROBCAL by 5% and 3%, respectively. Using explanations is more effective than using probabilities alone. In addition, P-E+EXPLCAL also outperforms P-E+ZHANG ET AL. (2021), whose performance is on par with P-E+PROBCAL. This suggests the additional CLS information is not very helpful in this setting.

As the data size increases from 32 to 128, the performance of the explanation-based calibrator keeps improving notably, whereas the performance of probability-based calibrators nearly saturates at a data size of 96. The performance of FEW-SHOT(NN) with 128 training instances only improves the performance by 3.3%, compared to FEW-SHOT with 32 training instances. Choosing nearest neighbors as the shots, while being effective when having access to a large amount of data, is not helpful in the extreme data-scarce regime. Calibrating using explanations is an effective way of using a few extra data points that cannot fit in the prompt, which is a pitfall of standard in-context learning.

Finally, ROBERTA finetuned using 128 shots only achieves an accuracy of 54.9%, lagging the performance of GPT-3 based models. The limited training data size is insufficient for finetuning smaller language models like ROBERTA, but is sufficient for P-E+EXPLCAL to be effective.

4.4.4 Calibrating HOTPOTQA

Setup For the HOTPOTQA dataset, our calibration takes the form of tuning the confidence scores of the predicted answers to better align them with the correctness of predictions. These confidence scores can be used in a "selective QA" setting (Kamath et al., 2020), where the model can abstain on a certain fraction of questions where it assigns low confidence to its answers. We use the *area under coverage-accuracy curve* (AUC) as in Chapter 3 to evaluate how well a model is calibrated as in past literature (Kamath et al., 2020; Chen et al., 2021a; Zhang et al., 2021; Garg and Moschitti, 2021). The curve plots the average accuracy with varying fractions (coverage) of questions being answered (examples in Figure 4.5). For

Table 4.4: AUC scores (mean$_{\text{std dev}}$) on HOTPOTQA under different data conditions. **L** and **E** denotes the number of label annotations and explanation annotations, respectively. Explanation-based calibration successfully improves the performance on top of prompting with explanations.

w/o Explanation	**6L**	**32L**	**64L**
FEW-SHOT	$59.6_{2.4}$	–	–
FEW-SHOT(NN)	–	–	$61.3_{0.9}$
w/ Explanation	**6L+6E**	**32L+6E**	**64L+6E**
E-P	$\mathbf{64.4}_{2.9}$	–	–
E-P+EXPLCAL	–	$\mathbf{66.0}_{3.9}$	$\mathbf{68.8}_{3.0}$
E-P+ZHANG	–	$65.6_{3.9}$	$66.1_{3.2}$

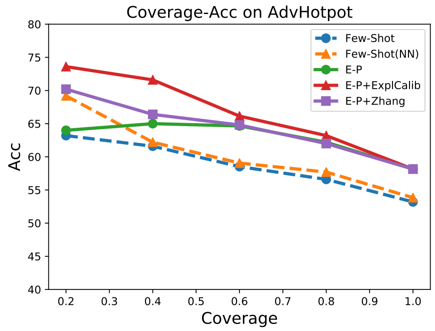


Figure 4.5: Coverage-Acc curves of various methods on HOTPOTQA. E-P+EXPLCAL is better calibrated compared to uncalbrated E-P as well as the other approaches.

any given coverage, a better calibrated model should be able to identify questions that it performs best on, hence resulting a higher AUC.

We experiment with training data set sizes of 6, 32, and 64. We report the results averaged from 5 trials using different training sets. For HOTPOTQA, we calibrate E-P, which is shown to be more effective than P-E in this setting (Section 4.2.4).

Results We show the AUC scores in Table 4.4. By leveraging explanations, E-P+EXPLCAL successfully achieves an AUC of 68.8, surpassing both FEW-SHOT by 7 points and E-P by 4 points. We note this is a substantial improvement, given that the upperbound of AUC is constrained by the accuracy of the answers and cannot reach 100. Figure 4.5 shows the coverage-accuracy curves of various methods averaged across the 5 training runs. E-P+EXPLCAL always achieves a higher accuracy than its uncalibrated counterpart, E-P, under a certain coverage, and the gap is especially large in the most confident intervals (coverage ¡ 50%). E-P+ZHANG ET AL. (2021) is able to calibrate the predictions on this dataset, but still lags our explanation-based calibrator, E-P+EXPLCAL.

In addition, the explanation-based calibrator can be effective with as few as 32 examples. This is because there are only two parameters (the probability of predicted answer and the explanation-based factor) in the calibrator, which can be easily learned in this few-shot setting. Comparing E-P+EXPLCAL against FEW-SHOT(NN), using nearest neighbors in the prompt is also able to improve the performance compared to using a fixed set of shots (FEW-SHOT), yet our lightweight calibrator can better utilize such a small amount of data, and learn to distinguish more accurate predictions based on the explanations.

4.5 Related Work

Our investigation is centered around in-context learning (Brown et al., 2020), which has garnered increasing interest since the breakthrough of various large pretrained language models. Recent work has been devoted to studying different aspects of in-context learning, including its wayward behaviors (Min et al., 2022; Webson and Pavlick, 2022) and approaches to overcome them (Zhao et al., 2021), whereas our exploration focuses on using explanations.

The utility of explanations for few-shot in-context learning has also been discussed concurrently (Nye et al., 2021; Wei et al., 2022c; Marasović et al., 2022; Chowdhery et al., 2022; Lampinen et al., 2022; Wiegreffe et al., 2022), especially in symbolic reasoning tasks. We differ in that we study more free-form explanations in tasks (QA and NLI, specifically) focusing on textual reasoning over provided contexts. Furthermore, our work focuses on the nature of the explanations generated by LLMs, which are found to be unreliable. Regarding our use of calibration, similar ideas of explanation-based performance estimation have been applied to other tasks (Rajani and Mooney, 2018) as well as in Chapter 2 and Chapter 3, but the method of this chapter relies on the free-text explanations generated by the model instead of interpretations obtained through post-hoc interpretation techniques.

More broadly, how to use explanations in various forms (textual explanation, highlights, etc.) to train better models is a longstanding problem (Zaidan et al., 2007). Past work has built a series of pipeline models that first generate the explanations and then make predictions purely based on the generated explanations (Wiegreffe et al., 2021; Zhou and Tan, 2021; Chen et al., 2022a). Prior research has also explored using explanations as additional supervision to train joint models (Hancock et al., 2018; Dua et al., 2020; Lamm et al., 2021; Stacey et al., 2022). Another line of work seeks to align the reasoning process of a trained model with the explanations, which is typically done by interpreting a prediction post-hoc through explanation techniques and optimizing the distance between the obtained explanation and ground truth explanation (Liu and Avci, 2019; Rieger et al., 2020; Plumb et al., 2020; Erion et al., 2021; Yao et al., 2021). These aforementioned methods all update the model parameters and typically require a considerable amount of explanation annotations to be effective. By contrast, our setting treats language models as pure black boxes and only requires few-shot explanations.

Lastly, our work connects with recent research focused on calibrating the predictions of LLMs Kadavath et al. (2022); Zhao et al. (2023); Huang et al. (2024); Zhu et al. (2023). These studies primarily aim to align the confidence scores of model predictions with the actual probability of their correctness, without altering the predictions themselves. Our approach differs in that we leverage explanations to adjust the relative confidence scores across different predictions, reranking candidate predictions to enhance the accuracy of the model's outputs.

4.6 Discussion & Conclusion

Caveats and Risks of Explanations from Large Language Models Our analysis suggests that LLMs' internal "reasoning" does not always align with explanations that it generates, as shown by our consistency results. More concerning, the explanations might not be factually grounded in the provided prompt. This shortcoming should caution against any deployment of this technology in practice: because the explanations are grammatical English and look very convincing, they may deceive users into believing the system's responses even when those responses are incorrect. Section 6 of Bender et al. (2021) discusses these risks in additional detail. The fact that language models can hallucinate explanations is also found in other work (Zhou and Tan, 2021). This result is unsurprising in some sense: without sufficient supervision or grounding, language models do not learn meaning as distinct from form (Bender and Koller, 2020), so we should not expect their explanations to be strongly grounded.

We have shown that even explanations which don't lead to accuracy gains can still be useful for calibration. However, the lexical overlap feature we use here is a weak signal of explanation correctness (see the example in Figure 4.1). Strong enough entailment models should theoretically be able to perform this task and work across a range of tasks without fine-tuning. This explanation assessment model can even be a language model itself trained for this particular propose to approach the verification tasks for a given domain by in-context learning.[9]

Conclusion We have explored the capabilities of LLMs in using explanations in in-context learning for textual reasoning. Through our experiments with four LLMs and on two QA datasets and an NLI dataset, we find that simply including explanations in the prompt does not always improve the performance of in-context learning. Our manual analysis demonstrates that LLMs tend to generate nonfactual explanations when making wrong predictions, which can be a useful leverage to assess the correctness of the predictions. Lastly, we showcase how to use explanations to build lightweight calibrators, which successfully improve InstructGPT's in-context learning performance across all three datasets.

[9] Since the release of an early published version of this chapter (Ye and Durrett, 2022b), there has been work that uses LLMs to evaluate the confidence of their predictions Kadavath et al., 2022, and more relevant to our work, the validity of reasoning chains Xie et al. (2023). As LLMs continue to grow in capability, we believe this represents a promising direction for future research.

Chapter 5: Explanation Selection for Chain-of-Thought Prompting[1]

5.1 Introduction

In the previous chapter, we show including *explanations* can potentially boost the prompting performance on a diverse of reasoning tasks, but sometimes may only lead to small to moderate improvements. Constructing effective explanations often require manual engineering (Wei et al., 2022c; Zhou et al., 2022a) to reach their full potential; past work has demonstrated that different combinations of explanations can lead to widely varying model performance (Wang et al., 2022b). These explanations are typically written in natural language (Madaan and Yazdanbakhsh, 2022; Ye et al., 2023b; Wang et al., 2023) and there are naturally many variants to explain the answer to a single question. Explanations in standard datasets written by crowdworkers may not be optimal, and even expert "prompt engineers" may not be able to easily elicit the best behavior.

This chapter studies the problem of optimizing explanations for better downstream performance on textual reasoning tasks. Inspired by recent work that bootstraps LLMs to improve reasoning (Zelikman et al., 2022; Huang et al., 2022), we propose an approach that can bootstrap a set of seed explanations (e.g., crowdworker annotated explanations) using an unlabeled development data set. As shown in Figure 5.1, we first prompt LLMs to construct alternative candidate explanations from the seed explanations. We then search over possible combinations of candidate explanations to find a combination that has high accuracy on the development set, which is silver-labeled using seed explanations.

Evaluating one candidate combination of explanations requires inference over the development set to compare against the silver labels. Given the cost of running LLMs,

[1] An early version of this chapter has been published in Ye and Durrett (2023). Xi Ye is the first author of Ye and Durrett (2023), where he developed the research idea with other author(s), implemented the framework, designed and performed the experiments and analysis, and wrote the paper.

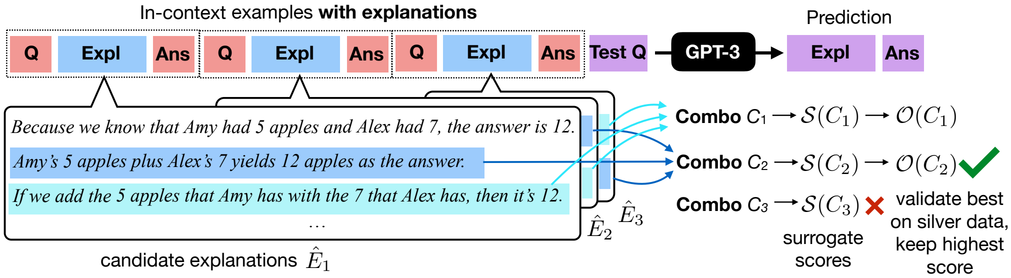

In-context examples **with explanations**
Q
Expl
Ans
Q
Expl
Ans
Q
Expl
Ans
Test Q
GPT-3
Prediction
Expl
Ans
Because we know that Amy had 5 apples and Alex had 7, the answer is 12.
Amy's 5 apples plus Alex's 7 yields 12 apples as the answer.
If we add the 5 apples that Amy has with the 7 that Alex has, then it's 12.
…
$\hat{E}_2$
$\hat{E}_3$
candidate explanations $\hat{E}_1$
Combo $C_1 \rightarrow \mathcal{S}(C_1) \rightarrow \mathcal{O}(C_1)$
Combo $C_2 \rightarrow \mathcal{S}(C_2) \rightarrow \mathcal{O}(C_2)$ ✓
Combo $C_3 \rightarrow \mathcal{S}(C_3)$ ✗
surrogate scores
validate best on silver data, keep highest score

Figure 5.1: Optimizing explanations given a candidate set. We generate candidate explanations in a leave-one-out fashion (not shown), prioritize combinations of explanations using a surrogate score $\mathcal{S}$, then evaluate them on silver data to optimize accuracy.

evaluating a large number of candidates is impractical. We propose a two-stage approach to efficiently search over potentially high-scoring combinations. We first evaluate each candidate explanation *in isolation* based on silver accuracy on the development set or the log likelihood on the few-shot training exemplar set. Scores of these individual explanations can be combined to compute scores of combinations, which gives a proxy of that combination's performance against silver set. We then can allocate our computation budget to evaluate better-performing candidate combinations based on the proxy metrics.

We apply our approach to optimize explanations on four datasets: GSM, ECQA, E-SNLI, and STRATEGYQA, covering a spectrum of textual reasoning tasks. Across the four datasets, our approach is able to find explanations that achieve 4% higher accuracy on average compared to initial seed explanations. We also show our proxy metrics can effectively approximate the downstream performance of combinations, and thus allow prioritizing search over better-performing explanations.

To summarize, our contributions are: (1) We propose a framework for optimizing explanations for in-context learning by optimizing over combinations of explanations. (2) We show that pseudo-labeling an unlabeled dataset can be used to evaluate such combinations. (3) We propose two proxy metrics to prioritize exploring better combinations given a limited computation budget.

5.2 Problem Formulation

5.2.1 Problem Statement

Following the standard chain-of-thought setting (Wei et al., 2022c), we assume access to a set of *exemplars* (input-output pairs) $T = \{(q_i, a_i)\}_{i=1:K}$ and *seed explanations* $\tilde{E} = \{\tilde{e}_i\}_{i=1:K}$ annotated for each exemplar in T (one per exemplar). In addition to T, some of our approaches assume access to an *unlabeled development set* V that only includes the inputs, i.e., $V = \{q_i\}_{i=1:M}$. Let θ be the parameters of an LLM.

Our goal is to find an explanation set $E = \{e_i\}_{i=1:K}$ that leads to the best accuracy. Each $e_i \in \Sigma^*$ is a natural language explanation expressed in the subword vocabulary Σ of the pre-trained language model. Past work has optimized many aspects of the in-context learning process, for example, the verbalization of prompts (Deng et al., 2022; Zhang et al., 2022b), exemplar selection (Ye et al., 2023b), and exemplar order (Lu et al., 2022), whereas our work focuses on optimizing the format of explanations in this particular way.

Because we assume a very small number of training examples, all of which are going to be included in the prompt, our notion of optimization (our "training objective") cannot rely on maximizing the likelihood of labeled training data. As we discuss in future sections, we will explore both likelihood-based measures as well as accuracy against pseudo-labeled versions of V. These objectives are also expensive to evaluate using LLMs, so we will operate under an additional constraint of cost in our methods.

Candidate explanations Directly searching over the combinatorial explanation space of E is intractable. Practically, we constrain the space of each e_i by selecting each from a *candidate explanation set* $\hat{E}_i = \{\hat{e}_i^{(1)} \dots \hat{e}_i^{(|\hat{E}_i|)}\}$, where each $\hat{e}_i^{(j)}$ denotes a candidate explanation associated with each exemplar q_i. The candidate explanation sets $\hat{E}_1 \dots \hat{E}_K$ can be generated by the LLM using a set of manually annotated seed explanations annotated by human $\tilde{E} = \{\tilde{e}_i\}_{i=1:K}$. That is, we use the exemplar set T and the seed sets $\tilde{E}$ excluding $(q_i, \tilde{e}_i, a_i)$ to prompt the LLM and draw N (40 in our implementation) samples for $\hat{E}_i$:

	Min	Avg	Max	Seed
GSM	57.7	61.8	66.0	61.9
ECQA	72.7	76.1	78.6	74.9
E-SNLI	60.3	72.3	80.1	71.8
STRATEGYQA	69.8	73.8	76.5	74.0

Table 5.1: Statistics of the performance of 16 different random combinations of explanations on 4 datasets and the performance of the seed explanations from crowdworkers. All tasks show substantial variation in performance.

$$(\hat{e}, \hat{a}) \ \sim p(e, a_i \mid \{(q_j, \tilde{e}_j, a_j)\}_{j=1:K \wedge j \neq i}, q_i; \theta) \tag{5.1}$$

Put another way, we use a leave-one-out approach to sample explanations and answers for each example using chain-of-thought prompting with $K-1$ examples. We reject any samples that do not have the correct answer for the example.

A combination C is a set of $\{e_i\}$ that contains one explanation e_i from the candidate explanation set $\hat{E}_i$, i.e., $C = \{e_i\}_{i=1:K} \wedge \forall i, e_i \in \hat{E}_i$. Now we can restate our problem: our goal is to find an explanation combination C that maximizes the accuracy when evaluating on test data.

5.2.2 Performance Varies Across Explanations

To illustrate the potential of our approach, we briefly analyze how using different explanations, for the same set of exemplars, can impact the downstream performance. As mentioned earlier, we generate candidate explanation sets according to Eq (5.1). Concretely, we use temperature scaling of 0.7 and sample 40 completions for each q_i, only retaining an $\bar{e}$ if it is paired with a correct answer $\bar{a} = a_i$. Note that for different q_i, we may find varying number of valid $\bar{e}$ (ranging from 0 to 40). We keep at most 8 for each q_i to save the search cost. We also include the seed explanations in the candidate explanation sets.

For each dataset, we randomly sample 16 combinations using the augmented can-

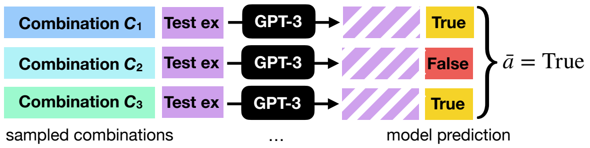


Figure 5.2: Silver labeling of unlabeled test example given several sampled combinations. This example is for a binary task with True or False labels (e.g., StrategyQA).

didate explanation sets, and report the statistics of the performance in Table 5.1. We see substantial variance in performance with different C: the average gap between the maximum performance and minimum performance exceeds 5% and is as large as 20% (on E-SNLI). In addition, the performance of seed explanations annotated by crowdworkers (SEED in Table 5.1) largely lags the best possible explanations, indicating substantial headroom for improvement.

5.3 Method Overview

Having candidate explanations for each question, we have reduced the search space from exponential in the vocabulary size to merely N^K. We then search over possible combinations of explanations. We describe our method for scoring combinations and the constraints under which our search takes place.

Pseudo-labeling development set We do not assume access to labeled examples beyond the K few-shot examples provided. However, we can take advantage of unlabeled data in V. We use a *pseudo-labeling* approach to derive labels for V following past work (Wang et al., 2022c). This approach is depicted in Figure 5.2; given $q \in V$, we sample random combinations of explanations to get predictions and use the majority-voted answer as the pseudo label $\hat{a}$:

$$\hat{a} = \arg\max_{a} \sum_{C=\{e_i\}} \mathbb{1}[a = \arg\max_{\bar{a}} p(\bar{a} \mid \{(q_i, e_i, a_i)\}_{i=1:K}, q; \theta)]$$

We now use the accuracy against the silver label as a surrogate objective $\mathcal{O}$, searching for C that maximizes accuracy with respect to the $\hat{a}$:

$$\mathcal{O}(C) = \operatorname*{arg\,max}_{C=\{e_i\}_{i=1:K}} \sum_{q_j \in V} \mathbb{1}[\hat{a}_j = \operatorname*{arg\,max}_{\bar{a}} p(\bar{a} \mid \{(q_i, e_i, a_i)\}_{i=1:K}, q_j; \theta)].$$

Searching over combinations One further complicating factor is that evaluating a combination C using $\mathcal{O}$ is expensive, as it requires running inference over the development set. We measure the computation budget B by the number of combinations needed to be scored using $\mathcal{O}$.

A naive approach is to randomly select B combinations to search, but this is inefficient. We propose additional surrogate metrics $\mathcal{S}$ to serve as a proxy for $\mathcal{O}$ for scoring combinations. We design $\mathcal{S}$ so that it can cost-efficiently score all combinations, with high $\mathcal{S}(C)$ indicating a combination C likely to obtain high $\mathcal{O}(C)$ score. In this way, $\mathcal{S}$ can be used to propose promising candidate combinations, only a few of which are scored using the actual objective $\mathcal{O}$ to save search budget.

5.4 Proxy Metrics for Finding Promising Combinations

Owning to the high cost, we only evaluate a small number (tens of combinations) of combinations against development set using $\mathcal{O}$ (Eq (5.3)). We first extract a set of promising combinations according to two proxy metrics, then evaluate those using our silver data.

5.4.1 One-shot Silver Accuracy

To optimize the silver accuracy of a combination of explanations (our objective $\mathcal{O}$), we hypothesize that *the prediction of a combination can be approximated with the prediction of each explanation used one-shot.* That is, we expect $p(a \mid \{(q_i, e_i, a_i)\}_{i=1:K}, q; \theta)$ to be higher when $\sum_{i=1:K} p(a \mid (q_i, e_i, a_i), q; \theta)$ is higher. We draw this hypothesis based on recent work on example selection for ICL, which shows that combining examples that individually perform well will yield better performance from the combination (Ye et al., 2023b; Rubin et al., 2022).

We define the average one-shot silver accuracy as a proxy metric $\mathcal{S}_{\text{OSAcc}}$:

$$\mathcal{S}_{\text{OSAcc}}(C = \{e_i\}_{i=1:K}) = \sum_{i=1:K} \sum_{q_j \in V} \mathbb{1}[\hat{a}_j = \arg\max_{\bar{a}} p(\bar{a} \mid (q_i, e_i, a_i), q_j; \theta)]$$

By computing the one-shot silver performance for $\forall \hat{e}_j^{(i)} \in \hat{E}^{(i)}$ for $\forall i = 1 : K$, we can efficiently compute the proxy metric $\mathcal{S}_{\text{OSAcc}}$ for any combination C.[2]

5.4.2 One-shot Log Likelihood

Besides using silver accuracy, another principle is to optimize the held-out log likelihood of the exemplar set:

$$\sum_{j=1:K} \log p(a_j \mid \{(q_i, e_i, a_i)\}_{i=1:K \wedge i \neq j}, q_j; \theta).$$

We apply a similar hypothesis and use the one-shot performance $\sum_{i=1:K \wedge i \neq j} p(a_j \mid (q_i, e_i, a_i), q_j; \theta)$ as the surrogate of $p(a_j \mid \{(q_i, e_i, a_i)\}_{i=1:K \wedge i \neq j}, q_j; \theta)$. We can then score a candidate combination by:

$$\sum_{j=1:K} \sum_{i=1:K \wedge i \neq j} \log \sum_{e} p(a_j, e \mid (q_i, e_i, a_i), q_j; \theta).$$

Since summing over explanations is intractable, we approximate this sum using the single sample of e to estimate the one-shot performance, leading to:

$$\mathcal{S}_{\text{OSLL}} = \sum_{j=1:K} \sum_{i=1:K \wedge i \neq j} \log p(e_j, a_j \mid (q_i, e_i, a_i), q_j; \theta). \tag{5.2}$$

We can compute $\mathcal{S}_{\text{OSLL}}$ for any C by only computing all the pairwise probabilities, $p(e_j, a_j \mid (q_i, e_i, a_i), q_j; \theta)$, for $\forall e_i \in \hat{E}_i, e_j \in \hat{E}_j \forall i = 1 : K, j = 1 : K \wedge i \neq j$, which is computationally feasible. Note that this metric does not require a development set.

[2] While this involves NK evaluations on the silver set, note that these evaluations are one-shot and significantly less computationally expensive than using higher numbers of shots.

	GSM		ECQA		ESNLI		STRATEGYQA	
METRICS	MAX@8	MAX@16	MAX@8	MAX@16	MAX@8	MAX@16	MAX@8	MAX@16
NAIVE	65.1	66.0	78.6	78.6	79.5	80.1	76.2	76.5
$\mathcal{S}_{\text{OSAcc}}$	**66.4**	**67.0**	79.7	80.5	**80.4**	**81.2**	74.3	74.9
$\mathcal{S}_{\text{OSLL}}$	65.7	65.9	**80.2**	**80.6**	75.8	76.5	**77.1**	**77.4**

Table 5.2: Oracle maximum accuracies achievable with 8 or 16 candidate combinations using different selection strategies. Using log likelihood-based or silver accuracy-based proxy metrics can find more promising candidate combinations than random candidates.

5.4.3 Ensemble

We have described the two proxy metrics using either the unlabeled set V or the labeled few-show exemplars T. Our further analysis (which we will describe later in Section 5.4) shows the choice of the most effective metric is task-specific. We additionally propose a strategy, ENSEMBLE of the $\mathcal{S}_{\text{OSLL}}$ and $\mathcal{S}_{\text{OSAcc}}$. Specifically, we first construct two sets of combinations that are preferred by these two proxy metrics individually, and then select the best one, from the union of these two sets, according to $\mathcal{O}$.

5.5 Experimental Setup

5.5.1 Language Models

We primarily use `code-davinci-002` (Chen et al., 2021b), a state-of-the-art LLM API, throughout our experiments, given its strong performance on various reasoning tasks (Li et al., 2022b). In addition, we use `text-davinci-003` to verify the effectiveness of the proxy metrics. `code-davinci-002` is a base model, and `text-davinci-003` is an Instruct-series model fine-tuned to align with human preferences (Ouyang et al., 2022).

Inference We follow past work to employ *greedy decoding* (greedily selecting the most probable token autoregressively) (Wei et al., 2022c) or self-consistency decoding (sampling tens of outputs from LLMs via temperature scaling and using popularity voting to assign a label) (Wang et al., 2022c).

Cost Querying LLMs is computationally intensive. We aim to search for better explanations within a reasonable budget. Our evaluation of cost is based on the *number of tokens* processed by LLMs, including both tokens in the prompts and the tokens generated by LLMs. We further bucket the measurement of cost by the number of combinations C that are scored by $\mathcal{O}$, which involves processing $M(K+1)$ examples.

5.5.2 Datasets

We experiment with four datasets covering four distinct tasks, including:

- GSM (Cobbe et al., 2021) consists of grade school math questions. Each is paired with a human-written explanation for the answer.
- ECQA (Aggarwal et al., 2021; Talmor et al., 2019) contains multiple-choice questions which test models' commonsense knowledge.
- E-SNLI (Camburu et al., 2018) studies the task of natural language inference which is to classify the relation between a premise and a hypothesis.
- STRATEGYQA (Geva et al., 2021a) asks Yes-No questions requiring steps. The dataset does not have explanation annotations, but it provides facts (Geva et al., 2021a) which are supporting evidence (albeit noisy ones) for the answers, so we use them as explanations.

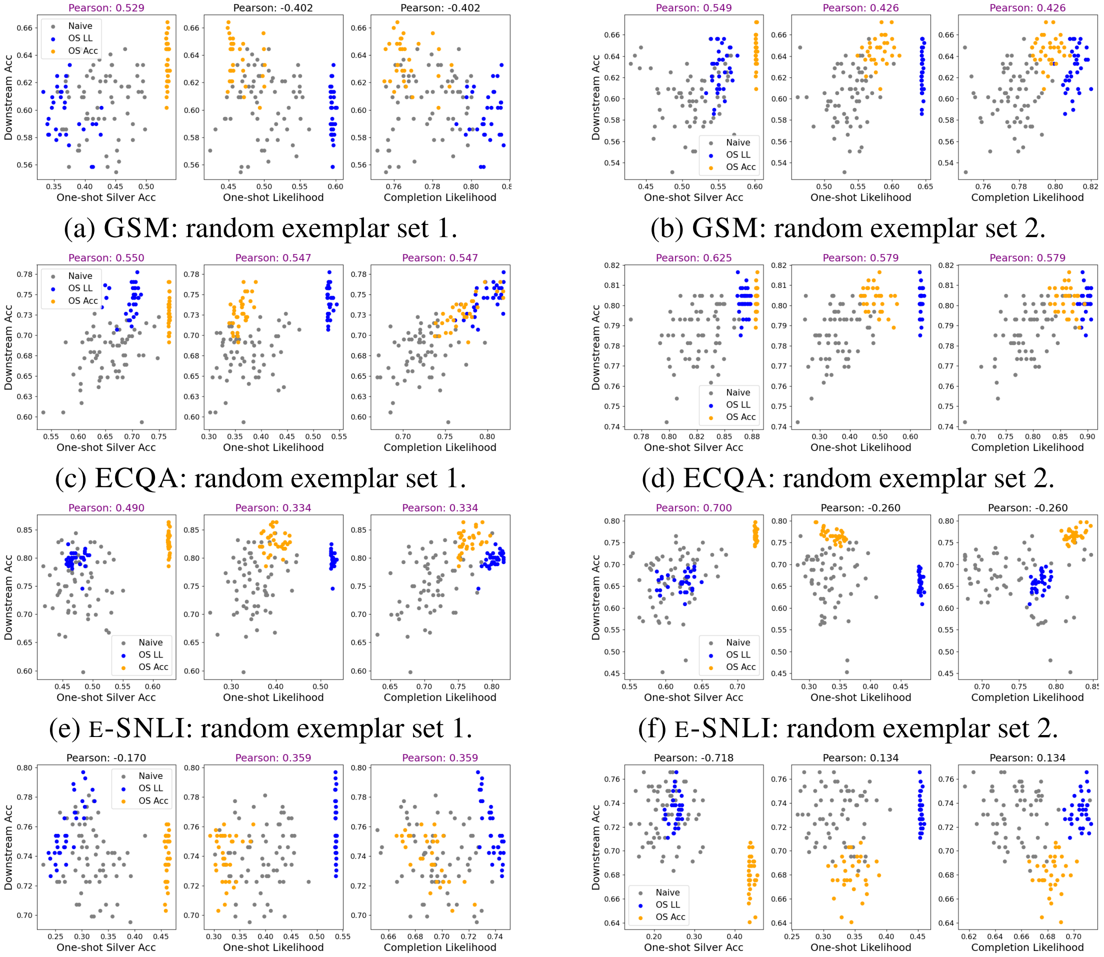
For each of the datasets, we choose prompt formats commonly used in past work (Wei et al., 2022c; Wang et al., 2022b). We use 8 exemplars in prompts for GSM, ECQA, and STRATEGYQA, and 9 exemplars (3 for each class) for E-SNLI, as sing more exemplars would not lead to further performance gains.

5.6 Effectiveness of Proxy Metrics

Before showing the results of the complete system, we first present experiments for verifying the effectiveness of the two proxy metrics. We evaluate them on the basis of the best oracle accuracy on a small (gold) labeled test set that we can reach using the top-X candidates, referred to as MAX@X, ranked by $\mathcal{S}_{\text{OSAcc}}$ or $\mathcal{S}_{\text{OSLL}}$. This gives an oracle upper bound for the performance that silver reranking via $\mathcal{O}$ can yield.

Setup We compare our metrics against a baseline which randomly scores combinations (NAIVE). We mainly use `code-davinci-002` for this experiment. For $\mathcal{S}_{\text{OSAcc}}$, we silver-labeled 256 randomly drawn development with 48 samples of combinations. For each dataset, we experiment with four different exemplar sets T to control for randomness and report the average number.

Results Table 5.2 shows the maximum reachable performance within 8 (Max@8) and 16 (Max@16) candidate combinations. For each dataset, using one of our metrics can find more promising candidate combinations than randomly proposed candidates. Among the



(a) GSM: random exemplar set 1.

(b) GSM: random exemplar set 2.

(c) ECQA: random exemplar set 1.

(d) ECQA: random exemplar set 2.

(e) E-SNLI: random exemplar set 1.

(f) E-SNLI: random exemplar set 2.

(g) STRATEGYQA: random exemplar set 1.

(h) STRATEGYQA: random exemplar set 2.

Figure 5.3: Gold test set accuracy (y-axis) vs. various surrogate proxy scores for explanation sets. Points of three different colors denote combinations selected using three metrics. Generally, there is a positive correlation between $\mathcal{S}_{\text{OSAcc}}$ (also $\mathcal{S}_{\text{OSLL}}$) and performance on these datasets.

top 16 combinations, combinations preferred by $\mathcal{S}_{\text{OSAcc}}$ can achieve better performance than randomly selected combinations by 1.0%, 0.9%, and 1.4% on GSM, ECQA, and E-SNLI, respectively. $\mathcal{S}_{\text{OSLL}}$ is the most effective strategy on ECQA, and STRATEGYQA, surpassing NAIVE by 2.0% and 0.9% on the basis of 16 candidate combinations. We do not find one metric that consistently gives the best performance.

Proxy metrics vs downstream accuracy In Figure 5.3, we show a series of graphs for intuitive understanding of how the proxy metrics relate to the downstream accuracy.

Each group of graphs shows the downstream accuracy vs. the surrogate proxy scores of combinations preferred by different metrics. For each dataset, we show two groups of graphs for two different exemplar sets out of four. Each group contains three graphs with different values on the x-axis. The first graph of a triple shows $\mathcal{S}_{\mathrm{OSAcc}}$ on the x-axis and the second one shows one-shot likelihood on the exemplar set (positively correlates with $\mathcal{S}_{\mathrm{OSLL}}$). In addition to the two proxy metrics, we show the completion likelihood on the third graph (probability of the predictions on the development set).

We show that the two surrogate scores we define mostly positively correlate with the downstream accuracy. $\mathcal{S}_{\mathrm{OSAcc}}$ (left) works uniformly well except on STRATEGYQA. $\mathcal{S}_{\mathrm{OSLL}}$ works well except for Figure 5.3a from GSM and Figure 5.3f from E-SNLI. In particular, on ECQA, both of them highly positively correlate with the downstream accuracy. Furthermore, we show the candidate combinations preferred by our proxy metrics lead to, in most cases, better likelihood on the development set (third graph in each triple), which indicates these combinations are more "optimized" for a specific task; past work suggests that better likelihood generally correlates with better downstream performance (Gonen et al., 2022).

5.7 Effectiveness of Framework

5.7.1 Main Results

We now test the effectiveness of the full framework. We mainly compare the performance of the explanations optimized via our approach against (1) the ZERO-COT approach (Kojima et al., 2022) (not using any human provided explanations) and (2) using seed explanations. In addition, we derive two baselines from past work on constructing effective explanations for ICL, which also select potentially better explanations from candidate explanations. Recall that $\hat{E}_i = \{\hat{e}_i^{(1)} \dots \hat{e}_i^{(|\hat{E}_i|)}\}$ is the candidate explanation set for q_i, our baselines include (1) BESTLEN that chooses the longest explanations (i.e., $\max_{\tilde{e}\in\tilde{E}} |\tilde{e}|$), as Fu et al. (2022) suggest using more complex CoTs leads to better performance for arithmetic reasoning, and (2) BESTPPL that chooses the explanation with the best perplexity (i.e., $\max_{\tilde{e}\in\tilde{E}} \mathrm{Perplexity}(a_i, \tilde{e}, q_i)$), as Gonen et al. (2022) suggest lower perplexity of prompts correlate with better performance. We note that these two baselines are not invented for optimizing explanations of given exemplars and are adapted to fit our setting. We refer to our optimization approach (based on the ENSEMBLE strategy) as OPTIMIZED.

Setup For all dataset sets, we experiment with 4 different exemplar sets as well as different unlabeled sets V of 256 randomly selected examples. We sample 48 combinations to silver label V. We constrain the computation budget B to be 50; this was the highest point feasible given limitations and was also where we found the silver accuracy ($\mathcal{O}$) to be nearly saturated. We note this budget has included the overhead for computing the proxy metrics as well as

	GSM	ECQA	E-SNLI	STRQA
ZERO-COT	30.9	61.2	49.7	55.1
SEED	62.6	77.0	75.2	71.3
BESTLEN	61.8	74.6	74.9	68.3
BESTPPL	63.4	79.4	76.5	69.0
OPTIMIZED	**66.0**	**83.0**	**82.8**	**71.6**

Table 5.3: The performance of optimized explanations against seed explanations and baselines derived from past work. Optimized explanations substantially outperform other approaches on GSM, ECQA, and E-SNLI.

the computation for scoring combinations using $\mathcal{O}$.

Results We show the performance of different approaches in Table 5.3. Overall, using our framework can find substantially better explanations measured by prompting performance compared to seed explanations. Without using any manually annotated explanations, the performance of ZERO-COT is far behind few-shot prompting using the seed explanations (SEED). Meanwhile, the explanations optimized using our framework outperforms the original seed explanations by 3.3%, 4.3%, and 7.1%, on GSM, ECQA, and E-SNLI, respectively. Choosing explanations with the lowest perplexity (BESTPPL) is able to marginally improve the performance on GSM, ECQA, and E-SNLI, compared to the seed set, but is consistently worse than our approach, and even leads to performance degradation on STRATEGYQA. As we are using 4 different random exemplar sets, we perform 4 groups of significance tests for different random trials.

5.7.2 Analysis

Self-consistency performance In addition to greedy decoding used in Table 5.3, we evaluate the performance of our optimized explanations under self-consistency decoding and compare against seed explanations. We vary the number of samples from 5 to 40, and show the results in Table 5.4. We note that the results are on a basis of one random exemplar set for each of the datasets, owing to the high computational cost of drawing tens of samples. As shown in Table 5.4, the optimized explanations consistently outperform the seed explanations under different numbers of samples. The gap is especially significant with smaller number of samples.

Results on other LLMs We mainly uses `code-davinci-002` in our experiments given its state-of-the-art ICL abilities. We also verify the effectiveness of our approach on

Num	Expl	GSM	ECQA	e-SNLI	StQA
5	Seed	70.4	79.8	80.0	72.9
	Optim	73.5	81.5	85.1	71.9
10	Seed	74.9	81.1	82.5	73.5
	Optim	78.9	82.1	85.5	73.1
20	Seed	79.1	81.2	83.7	74.4
	Optim	80.5	82.5	86.3	74.0
40	Seed	80.1	81.5	84.6	75.0
	Optim	81.2	82.5	87.2	75.4

Table 5.4: Performance of seed explanations and optimized (Optim) explanations using self-consistency decoding with varying number of samples.

	GSM	ECQA	e-SNLI	StrQA
Seed	58.2	74.3	81.0	67.6
Optimized	$61.3^{\Uparrow}$	$76.9^{\Uparrow}$	$82.8^{\uparrow}$	$69.4^{\Uparrow}$

Table 5.5: The performance of optimized explanations against seed explanations on `text-davinci-003` ($\Uparrow$ and $\uparrow$ denote significant improvements with p ¡ 0.05 and p ¡ 0.1, respectively). Our optimization approach is effective across LLMs.

`text-davinci-003`, an LLM finetuned to align with human feedback (Ouyang et al., 2022). We note that experiment with a smaller scale given the high cost and evaluate on one random set of exemplars instead of four. As shown in Table 5.5, applying our approach can also find better-performing explanations for all the datasets on `text-003`.

Generalizability of optimized explanations We investigate whether the performance improvements of our optimized explanations in a particular domain can generalize to other datasets with different distributions. Table 5.6 shows the performance of seed explanations and the optimized explanations from the GSM dataset (Optim-GSM) on the other arithmetic reasoning datasets, including SVAMP (Patel et al., 2021) and MAWPS (Koncel-Kedziorski et al., 2016). As suggested by the results, the optimized explanations achieve better performance compared to seed explanations on the out-of-domain datasets, which indicates that the performance improvements can generalize.

Results with reduced computation budget We expect search with our proxy metrics can still work well without high computation budget since they already extract potentially high-scoring combinations. We test a setting that uses a reduced computation budget. We set the

	SVAMP	SINEQ	SINOP	ADDSUB	MULARI
SEED	73.0	92.8	91.5	86.7	95.0
OPTIM-GSM	76.9	93.4	92.2	89.6	95.6

Table 5.6: Explanations optimized on the GSM dataset (OPTIM-GSM) achieve better performance on SVAMP and different settings of MAWPS compared to the seed explanations. The performance improvements of optimized explanations on one dataset can generalize to other out-of-domain datasets.

	GSM	ECQA	E-SNLI	STRQA
SEED	62.6	77.0	75.2	71.3
OPTIMIZED	64.5	81.2	81.5	71.0

Table 5.7: Results of searching with a reduced budget. Optimized explanations can still improve the performance upon the seed explanations.

budget to be 20. As seen in Table 5.7, with reduced budget, our framework can still improve the downstream performance compared to seed explanations by around 2.0%, 4.0%, and 6.0%, on GSM, ECQA, and E-SNLI, while maintaining performance on STRATEGYQA.

Failure analysis of proxy metrics In Section 5.6, we see that the $\mathcal{S}_{\text{OSLL}}$ and $\mathcal{S}_{\text{OSAcc}}$ do not always positively correlate with the performance on certain datasets. While we show such uncertainty can be handled by using an ensemble and scoring based on $\mathcal{O}$ we briefly analyze the failure of the two metrics for a better understanding of them.

In Table 5.2, $\mathcal{S}_{\text{OSAcc}}$ performs poorly on STRATEGYQA, yielding lower performance than the NAIVE strategy. The silver accuracy on this dataset is very poor: almost all one-shot accuracy is below 50% (see Figure 5.3g), worse than random guessing. One reason is that the binary nature of the task causes a single demonstration to be less suitable and representative than a single demonstration on more complex tasks like GSM. Under such circumstances, the averaged one-shot accuracy is no longer indicative of the full-prompt silver accuracy. On the other datasets, one-shot accuracy is meaningful (better than random guess), and the $\mathcal{S}_{\text{OSAcc}}$ correlates well with the full-prompt accuracy.

Furthermore, combinations scored highly by $\mathcal{S}_{\text{OSLL}}$ in Figure 5.3f are not better than random combinations in terms of downstream accuracy. Such combinations also lead to a mediocre completion likelihood, which is unusual as optimizing $\mathcal{S}_{\text{OSLL}}$ typically leads to the highest completion likelihood in other cases in Figure 5.3. We hypothesize this can be attributed to the distribution gap between the exemplar set and the test set. Since $\mathcal{S}_{\text{OSLL}}$ optimizes the log likelihood only based on the exemplar set, it might not generalize well to the test set under severe distribution shift, which is indicated by the suboptimal completion likelihood.

5.8 Related Work

We study prompting LLMs with chain-of-thought (Nye et al., 2021; Wei et al., 2022c; Shi et al., 2022) or textual explanations more generally (Marasović et al., 2022). Much of the past work focuses on exemplar selection in the presence of explanations (Fu et al., 2022; Ye et al., 2023b) or developing prompting methods for various reasoning tasks (Jung et al., 2022; Gao et al., 2022), which typically require manually engineered explanations. We focus instead on searching for better-performing explanations.

Our approach leverages data without explanation annotations. Similarly, prior work also explores the means of using few-show explanations together with data points without explanations annotations for improving downstream performance (Zelikman et al., 2022; Li et al., 2022b; Ye et al., 2023b; Li et al., 2022a; Wang et al., 2022a; Huang et al., 2022). Many of these techniques need a large amount of fully labeled data to train models used for generating explanations (Zelikman et al., 2022) or smaller models used as verifiers (Li et al., 2022b,a; Wang et al., 2022a), whereas our work only uses a small unlabeled set. There is also work on automatically constructing CoTs (Zhang et al., 2023) starting ZoTs (Kojima et al., 2022), which also requires a fully labeled dataset. In particular, Huang et al. (2022) also use LLMs to silver labeled data points for finetuning the LLMs; our work instead treats LLMs as black-boxes and searches for better explanations instead of tuning the parameters.

Our work also closely relates to prompt optimization. While experts can potentially engineer better prompts (Reynolds and McDonell, 2021; Mishra et al., 2022), such a process requires heavy manual effort. This has attracted growing interest on automated prompt engineering. One line of work requires interacting with gradients (Shin et al., 2020; Hu et al., 2021) or continuous embeddings (Sun et al., 2022a,b; Diao et al., 2022; Sun et al., 2023). Another line uses LMs as black-boxes (Prasad et al., 2022; Deng et al., 2022; Zhang et al., 2022b; Zhou et al., 2022c). However, this past work either optimizes over discrete templates (not applicable for the explanation optimization setting) or optimizes over string verbalizations (a search space too large for our setting).

5.9 Discussion & Conclusion

Limitations This chapter focuses on optimizing explanations solely for improving downstream prompting performance, without addressing other qualities of the explanations such as simulatability (Doshi-Velez and Kim, 2017), faithfulness (Subramanian et al., 2020; Wu and Mooney, 2019), and their utility in aiding human decision-making (Lipton, 2018). Our qualitative assessment of the optimized explanations does not indicate a decline in their quality, nor do they appear significantly better or more plausible as perceived by humans. In most cases, the optimization merely involves “mildly paraphrasing” the original explanations. The optimization objective here is designed to induce better test predictions in the final model. Part of the effects of this optimization may also be in the combination of the different explanations, so explanations may also be selected because they are more “compatible” with others in the final $\mathcal{O}$ ranking function. Future research could explore jointly optimizing the downstream prompting performance and the intrinsic quality of the explanations.

Conclusion We have presented an approach that can search for better-performing explanations for ICL starting from a set of seed explanations. Our approach first proposes promising candidate combinations of alternative explanations generated using LLMs, then finds explanation combinations using proxy metrics before using a silver-labeled validation set to select the best candidate. Our results highlight the substantial variance in the performance of different sets of explanations, paving the way for future work to further optimize explanations in this paradigm.

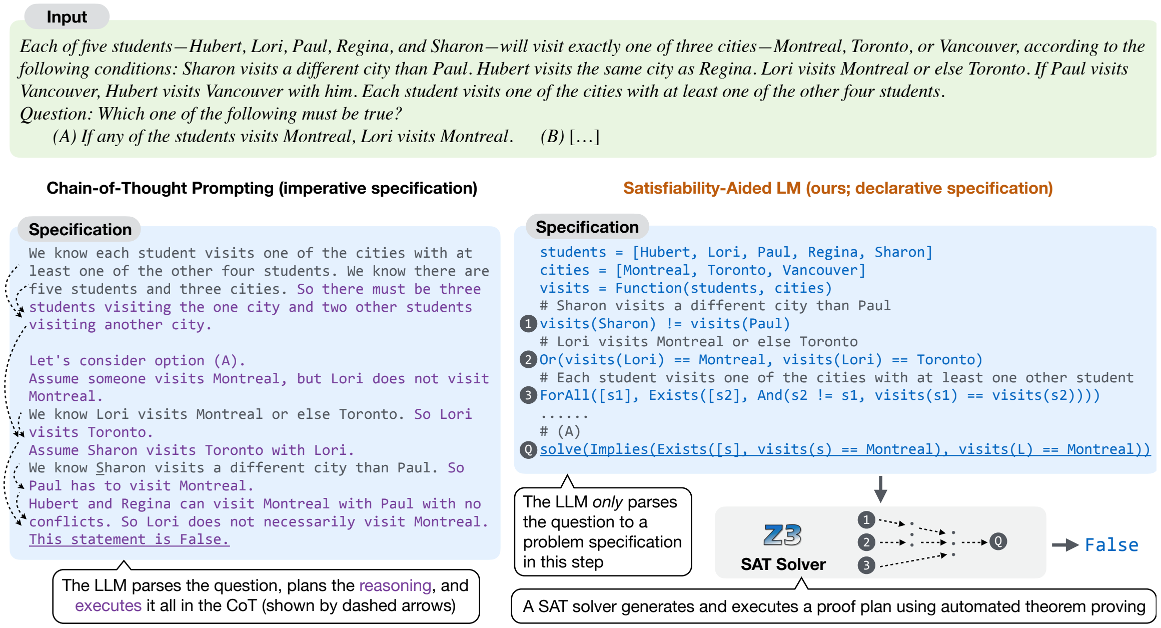
Chapter 6: Satisfiability-Aided Language Models Using Declarative Prompting[1]

6.1 Introduction

In Chapter 4 and Chapter 5, we focus on providing imperative style explanations such as chain-of-thought (CoT) (Wei et al., 2022c) to aid LLMs in reasoning, which enable LLMs to follow a sequence of reasoning steps before making a prediction. This is effective on various multi-step reasoning tasks, especially those with fixed forward reasoning procedures, e.g., concatenating the last letters of several words. However, imperative explanations can fall short when scaling to problems that involve intensive computation (Gao et al., 2023) or long sequences of reasoning steps (Creswell et al., 2023; Saparov and He, 2023; Ribeiro et al., 2023), especially as LLMs exhibit limited planning capabilities (Valmeekam et al., 2022; Liu et al., 2023a).

In this chapter, we show how we can use declarative formal specifications as explanations and how we can combine LLMs with symbolic solvers, which scales LLMs to complex problems requiring deep reasoning depth. Solving a complex reasoning problem involves three conceptual components: parsing a natural language description into a representation of the problem, deriving a plan to solve the problem, and executing that plan to obtain an answer. Recent work on improving CoT prompting focuses on fixing *execution errors* by augmenting LLMs with symbolic executors such as a Python interpreter, which leads to improved performance on arithmetic and symbolic reasoning tasks (Gao et al., 2023; Chen et al., 2022b; Lyu et al., 2023). However, CoT prompting (Wei et al., 2022c; Nye et al., 2021) and its executor-augmented successors (Gao et al., 2023; Chen et al., 2022b; Lyu et al., 2023) are oriented towards *imperative* solving procedures: a CoT or a program specifies the reasoning procedure as chained steps (Wei et al., 2022c; Gao et al., 2023)

[1] An early version of this Chapter has been published in Ye et al. (2023a). Xi Ye is the first author of Ye et al. (2023a), where he developed the research idea with other author(s), implemented the framework, designed and performed the experiments and analysis, and wrote the paper.



Input

Each of five students—Hubert, Lori, Paul, Regina, and Sharon—will visit exactly one of three cities—Montreal, Toronto, or Vancouver, according to the following conditions: Sharon visits a different city than Paul. Hubert visits the same city as Regina. Lori visits Montreal or else Toronto. If Paul visits Vancouver, Hubert visits Vancouver with him. Each student visits one of the cities with at least one of the other four students.
Question: Which one of the following must be true?
(A) If any of the students visits Montreal, Lori visits Montreal. *(B)* [...]

Chain-of-Thought Prompting (imperative specification)

Specification

```
We know each student visits one of the cities with at
least one of the other four students. We know there are
five students and three cities. So there must be three
students visiting the one city and two other students
visiting another city.

Let's consider option (A).
Assume someone visits Montreal, but Lori does not visit
Montreal.
We know Lori visits Montreal or else Toronto. So Lori
visits Toronto.
Assume Sharon visits Toronto with Lori.
We know Sharon visits a different city than Paul. So
Paul has to visit Montreal.
Hubert and Regina can visit Montreal with Paul with no
conflicts. So Lori does not necessarily visit Montreal.
This statement is False.
```

The LLM parses the question, plans the reasoning, and executes it all in the CoT (shown by dashed arrows)

Satisfiability-Aided LM (ours; declarative specification)

Specification

```
 students = [Hubert, Lori, Paul, Regina, Sharon]
 cities = [Montreal, Toronto, Vancouver]
 visits = Function(students, cities)
 # Sharon visits a different city than Paul
1visits(Sharon) != visits(Paul)
 # Lori visits Montreal or else Toronto
2Or(visits(Lori) == Montreal, visits(Lori) == Toronto)
 # Each student visits one of the cities with at least one other student
3ForAll([s1], Exists([s2], And(s2 != s1, visits(s1) == visits(s2))))
 ......
 # (A)
Qsolve(Implies(Exists([s], visits(s) == Montreal), visits(L) == Montreal))
```

The LLM *only* parses the question to a problem specification in this step

A SAT solver generates and executes a proof plan using automated theorem proving

Figure 6.1: Illustration of our Satisfiability-aided Language Modeling approach (right). We first parse an NL input into a declarative task specification (a set of logical constraints) using prompting (Section 6.3.1), then use a SAT solver to solve the problem (Section 6.3.2). The chain-of-thought strategy in prior work (left) yields imperative reasoning processes.

in the order of execution. While this is effective for problems whose natural language already provides a suitably clear "plan" for the reasoning, it only leads to limited success for reasoning problems like in Figure 6.1 that do not outline such a plan (Ribeiro et al., 2023). These problems often state a set of premises and constraints and ask questions that require sophisticated planning to deductively reason over the inputs, which is still challenging even for modern LLMs (Valmeekam et al., 2022).

Our work tackles both execution errors and, more importantly, *planning errors*. We propose SATisfiablity-aided Language Modeling (SATLM) using declarative prompting. The core idea is to cast a natural language (NL) reasoning problem as a satisfiability (SAT for short) problem. As shown in Figure 6.1 (right), given a problem in NL, we prompt an LLM to parse it into a SAT problem specification which consists of a set of logical formulas, then obtain the solution by invoking a SAT solver.[2] The LLM is specialized towards understanding the preconditions stated in the problem, while the solver is leveraged to plan out the reasoning procedure. In addition, the solver guarantees the correctness of execution, similar to the interpreter used in program-aided LMs (PROGLM).

We evaluate our approach on 8 datasets spanning 4 tasks, including arithmetic reasoning, logical reasoning, symbolic reasoning, and a regex synthesis task. Our SATLM consistently outperforms COT and PROGLM across all datasets, usually by a large margin. On GSM-SYS, SATLM outperforms PROGLM by a 23%; on GSM, SATLM achieves 84.8% with self-consistency decoding using few-shot prompting, equaling past work that uses the full training set and the same LLM (Li et al., 2022b; Ni et al., 2023). SATLM also sets a new SoTA on LSAT (Zhong et al., 2022), BOARDGAMEQA (Kazemi et al., 2023), and STRUCTUREDREGEX (Ye et al., 2020b).

Our analysis illustrates why the combination of SAT solver and declarative prompting is so effective. We find (1) program-aided LMs often make planning errors (e.g., manipulating equations incorrectly), which can be remedied by the SAT solver. (2) Forcing LLMs to explicitly state a declarative specification can even improve vanilla CoT prompting. (3) Our SATLM approach can abstain from making uncertain predictions if it parses a problem into an unsatisfiable or ambiguous specification, giving it even higher accuracy in the selective prediction setting (El-Yaniv and Wiener, 2010).

6.2 Overview

This work addresses the challenge of using LLMs to solve NL reasoning tasks. At a high level, an NL reasoning task is a natural language description of a collection of facts Φ (such as propositions or constraints) about some objects and a question Q related to these objects. The goal of the reasoning task is to find an answer to Q that can be deduced from

[2] Here, we use SAT solver to refer to any automated reasoning tool for checking the satisfiability of formulas in formal logic. Hence, "SAT solver" in this Chapter also includes first-order theorem provers and SMT solvers.

the information provided in Φ.

We conceptualize the general procedure for solving NL reasoning tasks in three steps: *parsing*, *planning*, and *execution*. We are given natural language input $x_{\texttt{test}} = (NL(\Phi), NL(Q))$ which describes both Φ and Q. Our first step is to parse this natural language into a predicted *task specification* $(\hat{\Phi}, \hat{Q})$, which is a *formal* description of the facts and the query.

Given $(\hat{\Phi}, \hat{Q})$, the planning step then involves determining a sequence of reasoning steps $[r_1, \ldots, r_n]$ beginning with the task specification and ending with the answer to the question. Each step involves invoking a function (e.g., arithmetic operator or logical operator) that produces intermediate results which can be utilized in subsequent steps. A plan can be formulated by an LLM with CoT prompting or by a symbolic solver as in our work here. Finally, we execute the plan systematically with either a symbolic executor (our method) or an LLM, returning the output of the last step, r_n, as the answer.

Our solution approaches the problem using exactly these three steps.

Parsing into declarative specification We prompt an LLM to generate a specification $s_{\texttt{test}}$ for $x_{\texttt{test}}$. Note that the translation from this description into the specification is not straightforward and cannot be done in a rule-based way for most tasks; Figure 6.4 shows some particularly complex examples involving commonsense reasoning. The specification $s_{\texttt{test}}$ is a sequence of interleaved NL statements and logical formulas (LF): $s_{\texttt{test}} = [z_1, \ldots, z_n]$ and $z_i \in \Sigma_{NL} \cup \Sigma_{LF}$, where Σ_{NL} and Σ_{LF} denote the space of natural language and logical formulas, respectively. We derive the formal specification $(\hat{\Phi}, \hat{Q})$ by taking all the z_i in Σ_{LF} from $s_{\texttt{test}}$. An example of the specification is presented on the right of Figure 6.1. Our specification is declarative since we do not explicitly generate the r_i from the LLM at this stage.

Planning and execution with a SAT solver Given the predicted formal specification $(\hat{\Phi}, \hat{Q})$, we wish to derive the final answer of the query $\hat{Q}$ from it. We say that a solution a is correct if $\hat{\Phi}$ logically entails $\hat{Q} = a$, denoted as $\hat{\Phi} \models \hat{Q} = a$. The key insight behind our work is to offload *both* the planning and execution steps to a SAT solver. Specifically, we use a SAT solver to find a satisfying assignment for a in the formula:

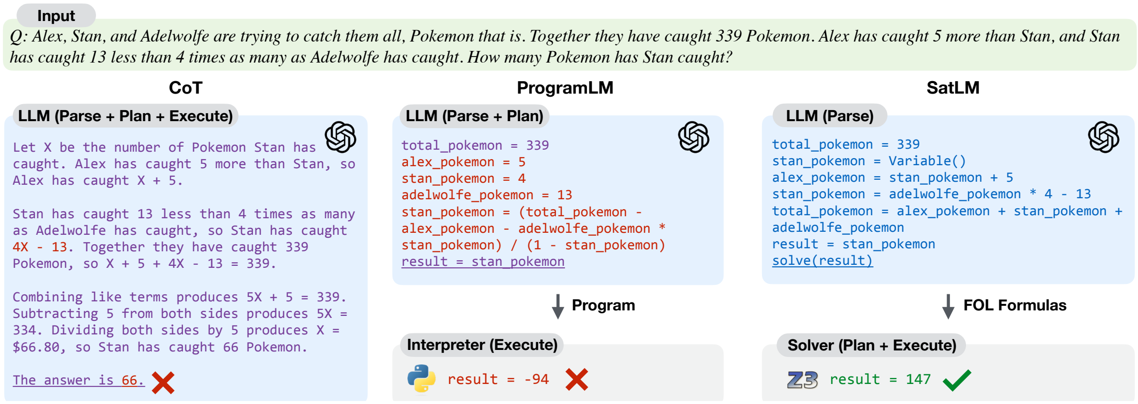
$$\forall V. \ (\hat{\Phi} \Rightarrow \hat{Q} = a)$$

where V denotes the set of all variables used in $\hat{\Phi}$ and $\hat{Q} \in V$ is a variable that corresponds to the solution. Note that the only free variable in this formula is a; hence, the assignment to a returned by the solver is the final answer to the reasoning problem.

The approach outlined above has two important strengths. First, because the SAT solver is *sound* (i.e., any assignment it produces satisfies the formula), the solution is correct by construction. Thus, assuming that the parsing is correct and $\hat{\Phi}$ and $\hat{Q}$ match Φ and Q, we have a proof that the solution is indeed correct. Second, the planning step is done internally to the solver, and the chain of reasoning steps $[r_1, \dots, r_n]$ can be obtained by asking the solver to produce a proof of the validity of the formula $\hat{\Phi} \Rightarrow \hat{Q} = a^*$ where a^* is the assignment produced by the SAT solver. All solvers we consider can produce such a proof of validity (e.g., in the form of a resolution refutation (Davis and Putnam, 1960)).

Comparison with prior LLM reasoning approaches Prior approaches to NL-based reasoning with LLMs can also be framed in the parse-plan-execute framework proposed above. In particular, the chain-of-thought paradigm (Nye et al., 2021; Wei et al., 2022c) uses LLMs to perform each of the three steps. Program-aided language models (Gao et al., 2023; Chen et al., 2022b; Lyu et al., 2023) combine the parsing and planning steps to use an LLM to derive a program that corresponds to the plan.[3] The final execution step is then performed by using the interpreter of the underlying programming language to derive the

[3] This is true for "faithful chain-of-thought" as well (Lyu et al., 2023). This Chapter describes a breakdown of the process into "translation" and "solving" stages, where the translation step corresponds to both our parsing and planning stages. The solver used in that approach for tasks like CLUTRR does not do additional planning, but merely executes the steps outlined in CoT. In addition, their approach uses Python for execution, whereas ours uses SAT and Z3 as the unifying solving framework.



Q: Alex, Stan, and Adelwolfe are trying to catch them all, Pokemon that is. Together they have caught 339 Pokemon. Alex has caught 5 more than Stan, and Stan has caught 13 less than 4 times as many as Adelwolfe has caught. How many Pokemon has Stan caught?

```
Let X be the number of Pokemon Stan has
caught. Alex has caught 5 more than Stan, so
Alex has caught X + 5.

Stan has caught 13 less than 4 times as many
as Adelwolfe has caught, so Stan has caught
4X - 13. Together they have caught 339
Pokemon, so X + 5 + 4X - 13 = 339.

Combining like terms produces 5X + 5 = 339.
Subtracting 5 from both sides produces 5X =
334. Dividing both sides by 5 produces X =
$66.80, so Stan has caught 66 Pokemon.

The answer is 66. ✗
```

```
total_pokemon = 339
alex_pokemon = 5
stan_pokemon = 4
adelwolfe_pokemon = 13
stan_pokemon = (total_pokemon -
alex_pokemon - adelwolfe_pokemon *
stan_pokemon) / (1 - stan_pokemon)
result = stan_pokemon
```

```
result = -94 ✗
```

```
total_pokemon = 339
stan_pokemon = Variable()
alex_pokemon = stan_pokemon + 5
stan_pokemon = adelwolfe_pokemon * 4 - 13
total_pokemon = alex_pokemon + stan_pokemon +
adelwolfe_pokemon
result = stan_pokemon
solve(result)
```

```
result = 147 ✓
```

Figure 6.2: Exemplar specifications for arithmetic reasoning problems generated by different approaches. CoT makes errors when parsing an equation; PROGLM produces an incorrect reasoning chain (both errors are highlighted in red). By only using the LLMs to generate declarative specifications and relying on a solver to handle the reasoning, SATLM generates the correct answer.

final answer. In contrast to these approaches, our work uses an LLM only to perform the parsing step, which is an easier problem for LLMs than planning.

We show a concrete example comparing CoT and PROGLM with our approach in Figure 6.2. CoT performs all three steps with the LLM. For instance, "*Alex has caught* $X + 5$" in the output corresponds to "*Alex has caught 5 more than Stan*" in the NL input (parsing). Later, CoT decides how to solve for the variable X with "*Combining like terms* ..." (planning). At the same time, it also derives the equation "$5X = 334$" directly in its generation (execution). However, CoT incorrectly uses the same X in the equation "$X + 5$" and "$4X - 13$", when it is supposed to be different. (Note that $4X - 13$ would be correct if Stan and Adelwolfe's roles in the corresponding NL clause were reversed.) By allowing the LLM to focus only on translation, we find a lower incidence of this kind of error, in addition to eliminating planning errors. Notably, planning errors are **not** addressed by PROGLM, which does not use programmatic manipulation at this stage. Different from PROGLM, SATLM only parses the information provided in the input question, passes the parsed formulas to a solver for both planning and execution, and obtains the correct result.

6.3 SAT-Aided Language Models using Declarative Prompting

6.3.1 Declarative Prompting

We use few-shot prompting to generate the specification $s_{\texttt{test}}$ for the test input $x_{\texttt{test}}$. Specifically, we include few-shot demonstrations $(x_i, s_i)_{i=1}^{k}$ in the prompt, append test input $x_{\texttt{test}}$ after the prompt, and let the LLM complete the specification for $x_{\texttt{test}}$, i.e., $s_{\texttt{test}} \sim p(x_{\texttt{test}} \mid x_1, s_1, \ldots, x_k, s_k)$.

We show an example specification for a logical reasoning task in Figure 6.1, and an example specification for an arithmetic reasoning task in Figure 6.2. Observe that in both examples, our SAT formulas (i.e., the logical formulas of $[z_1, \ldots, z_n]$ in Σ_{LF}) are written as code following Python syntax, while the natural language in Σ_{NL} is written using comment syntax. We found that including the language here as comments was useful to improve the fidelity of the translation. Our declarative prompts also use meaningful variable names and descriptive comments following the style of prompts in prior work (Gao et al., 2023; Lyu et al., 2023). Finally, we use Python rather than a specialized DSL to be more congruent with our models' pretraining data (Ouyang et al., 2022; Chen et al., 2021b).

6.3.2 Solving with a SAT Solver

SAT problem A SAT problem is a triple $\mathcal{P} = (\Phi, \mathcal{T}, Q)$ where Φ is a set of first-order logic formulas in some theory $\mathcal{T}$[4] and Q is the query of interest. We use $\texttt{Variable}(\mathcal{P})$ to denote the free variables in Φ. Q contains only variables in $\texttt{Variable}(\mathcal{P})$. An example SAT problem is $\mathcal{P} = (\{x + y = 3, x - y = 1\}, \mathcal{T}_E \cup \mathcal{T}_{\mathbb{Z}}, x - 2)$, where $\mathcal{T}_E \cup \mathcal{T}_{\mathbb{Z}}$ indicates that only equality and linear arithmetic operations on integers are allowed in the formulas.

Many NL reasoning tasks in the literature can be formulated as SAT problems and solved using an off-the-shelf solver. For **arithmetic reasoning**, the SAT formulas Φ are equations encoding the relationships between variables, and t specifies the target variable asked in the question (see Figure 6.1). For **logical reasoning**, Φ encodes preconditions

[4] The theory defines the meaning of some of the symbols used in the formula. For example, in the theory of linear arithmetic, axioms of the theory give meaning to operators like addition, less than, etc.

and t specifies the target statement posed by the question. We also show that symbolic reasoning, regex synthesis, and other problems involving reasoning over arrays or strings can be handled in this framework.

Unlike prior work such as Faithful CoT (Lyu et al., 2023) that uses task-specific formulations and task-specific solvers for different problem types, all the tasks in this Chapter are formulated as general SAT instances that can be solved by a single solver (as described later in this section).

Parsing NL to a SAT problem Recall that we obtain a specification $s_{\texttt{test}}$ from a test NL task $x_{\texttt{test}}$. To derive the SAT problem $\mathcal{P}_{\texttt{test}} = (\hat{\Phi}_{\texttt{test}}, \mathcal{T}_{\texttt{test}}, \hat{Q}_{\texttt{test}})$ from $s_{\texttt{test}}$, we extract the constraints $\hat{\Phi}_{\texttt{test}}$ and the target expression $\hat{Q}_{\texttt{test}}$ (marked by `solve` in our prompt) by taking all the z_i in Σ_{LF} of $s_{\texttt{test}}$. We identify the theory $\mathcal{T}_{\texttt{test}}$ by analyzing the formulas in $\hat{\Phi}_{\texttt{test}}$.

Solving the SAT problem Given the SAT problem $\mathcal{P}$, we invoke an automated theorem prover (such as the Z3 SMT solver (De Moura and Bjørner, 2008) used in our implementation) to obtain a model M that maps each free variable $v \in \texttt{Variable}(\mathcal{P})$ to a concrete value under theory $\mathcal{T}$. The final answer is obtained by substituting each free variable v_i in $\hat{Q}$ with $M[v_i]$. For example, given the problem $(\{x + y = 3, x - y = 1\}, \mathcal{T}_E \cup \mathcal{T}_{\mathbb{Z}}, x - 2)$, we ask the solver to find a solution to the constraint $x + y = 3 \wedge x - y = 1$ in the theory $\mathcal{T}_E \cup \mathcal{T}_{\mathbb{Z}}$, which yields $x = 2$ and $y = 1$. Then, to obtain the final answer, we substitute x by 2 in the target expression $x - 2$ to obtain the result $2 - 2 = 0$.

Feedback signals from the solver Given a set of $\hat{\Phi}$ specified in $\mathcal{P}$, the SAT solver will try to search for a satisfying assignment M which satisfies all constraint formulas in $\hat{\Phi}$. If the solver succeeds in finding such an assignment within a certain time limit, it will use M to evaluate the query $\hat{Q}$ and return the final result, otherwise it is a timeout. However, the solver may fail to find a solution for problematic $\mathcal{P}$ and provide feedback in one of the following types: (1) ***error in execution*** (ERROR) caused by invalid formulas (e.g., syntax errors) or time-out; (2) ***unsatisfiable formulas*** (UNSAT), caused by conflicting formulas in the $\hat{\Phi}$ (e.g. $\hat{\Phi} = \{x = y + 1, y = x + 1\}$) (no feasible solution); (3) ***ambiguous formulas*** (AMBIG), caused by the existence of multiple feasible solutions (e.g. $\hat{\Phi} = \{x = y + 1, x > 0\}$).

Unlike the executor used in PROGLM that can only detect errors in code execution, SAT solver can spot UNSAT and AMBIG in addition to ERROR. We show this unique characteristic allows our SATLM to abstain from potentially incorrect predictions much more effectively compared to PROGLM in the selective prediction setting (El-Yaniv and Wiener, 2010) (Section 6.4.4).

6.4 Experiments

6.4.1 Setup

Tasks Our work investigates 8 datasets covering 4 tasks, with a focus on arithmetic reasoning and logical reasoning tasks. For arithmetic reasoning, we use GSM (Cobbe et al., 2021), GSM-SYS, and ALGEBRA (He-Yueya et al., 2023). GSM-SYS is a special subset of GSM containing examples that are paired with human-annotated solutions involving systems of equations. For logical reasoning, we use LSAT (Zhong et al., 2022), BOARDGAMEQA (Kazemi et al., 2023), CLUTRR (Sinha et al., 2019), and PROOFWRITER (Tafjord et al., 2021). For BOARDGAMEQA, we report the average performance on the three data splits (depth 1 to depth 3).

For CLUTRR, we use exemplars requiring up to 3 intermediate steps but evaluate on test examples requiring up to 10 intermediate steps (Sinha et al., 2019), following past work (Lyu et al., 2023). For PROOFWRITER, we evaluate on the most challenging examples requiring depth-5 proofs (Tafjord et al., 2021). For symbolic reasoning, we use Colored Object (COLOR) from BIG-bench (et al., 2022) as an exemplar task. This task can be abstracted as finding elements in a list under certain constraints. We also evaluate on a regex synthesis dataset, STREGEX (Ye et al., 2020b), which requires synthesizing a regex give NL description. We cast this task into synthesizing the surface form (i.e., a string) of the target regex, and use SATLM to parse NL description into constraints over the string.

Baselines We compare SATLM against 3 baselines, including standard prompting (directly giving the answer), chain-of-thought prompting (COT), and executor-augmented LLMs (PROGLM). We do not compare to zero-shot baselines such as zero-shot CoT, which generally underperform few-shot CoT by a large margin on the tasks we investigate (Kojima et al., 2022).

For COT and PROGLM, we leverage prompts of existing work (Gao et al., 2023; Lyu et al., 2023; Creswell et al., 2023) whenever possible. For SATLM, we manually write prompts for the **same exemplar sets** used in COT and PROGLM to ensure a fair comparison. We note that some settings, such as PROGLM for LSAT, are not applicable.

Language Models & Decoding We conduct our main experiments and analysis on `code-davinci-002` (Chen et al., 2021b), a state-of-art LLM for code and code-adjacent tasks. We evaluate the performance with both greedy decoding and self-consistency decoding (Wang et al., 2022c). Following past work (Gao et al., 2023), we use 40 samples on all datasets except for LSAT, BOARDGAMEQA, and PROOFWRITER; we use 5 samples on these datasets involving long prompts and high computation cost. For COT and PROGLM, we use a temperature of 0.7; for SATLM, we use a higher temperature of 0.9, which we find to work better.

6.4.2 Main Results

Table 6.1 shows the performance of our approach compared to the baselines. In general, our SAT-aided approach outperforms both COT and PROGLM by a substantial margin except on GSM with greedy decoding. We perform significance tests via bootstrap resampling, and all improvements of SATLM over PROGLM are statistically significant ($p < 0.05$).

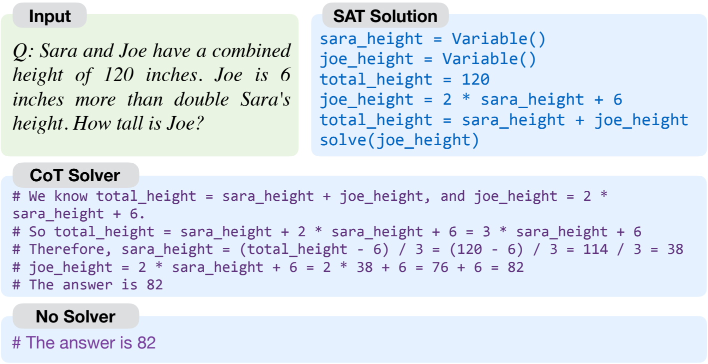
The first two columns show the performance on the GSM dataset. COT and

Table 6.1: Comparison of our approach (SatLM) against standard prompting (directly predicting the answer), CoT and ProgLM. Certain settings are not applicable (marked as −). With greedy decoding, SatLM outperforms CoT and ProgLM on all datasets by a substantial margin except for GSM, where it is on par with ProgLM. With self-consistency decoding, SatLM is consistently better than ProgLM, giving SoTA accuracy on LSAT and BoardgameQA.

	GSM-Sys	GSM	Alge	LSAT	Board	Clutrr	Proof	Color	Regex
					code-davinci-002 (greedy decoding)				
Standard	21.0	22.2	45.9	22.0	44.6	41.2	76.6	75.7	–
CoT	46.5	62.7	53.6	23.5	60.7	40.8	80.1	86.3	–
ProgLM	43.4	**72.7**	52.3	–	–	58.9	83.7	95.1	39.1
SatLM	**69.4**	71.8	**77.5**	**35.0**	**79.4**	**68.3**	**99.7**	**97.7**	**41.0**
					code-davinci-002 (self-consistency decoding)				
CoT	56.1	77.3	64.9	23.1	62.8	45.7	88.7	90.6	–
ProgLM	53.4	82.4	57.7	–	–	71.9	91.2	98.0	56.5
SatLM	**80.9**	**84.8**	**90.9**	**37.4**	**80.7**	**80.1**	**99.7**	**99.4**	**59.7**

ProgLM achieve much worse performance on GSM-Sys than on GSM, indicating that GSM-Sys is a challenging subset. On this subset, SatLM achieves 69.4% and 80.9% with greedy decoding and self-consistency decoding, surpassing both ProgLM and CoT more than by 20%. On the original GSM dataset, the SatLM model has a slightly lower accuracy than ProgLM with greedy decoding, but outperforms it with self-consistency decoding by 2.4%; we provide detailed analysis accounting for the differences later in this section. This self-consistency accuracy of 84.8% even exceeds recent work that uses the full training set with `code-davinci-002` (82.3% in DiVeRSe (Li et al., 2022b); 84.5% in Lever (Ni et al., 2023)). On Algebra, a challenging dataset of math problems extracted from algebra textbooks, SatLM also outperforms CoT and ProgLM by more than 20%.

On LSAT, Clutrr, ProofWriter, and Color, SatLM consistently achieves the best performance with either greedy decoding or self-consistency decoding. SatLM also sets the new SoTA on both LSAT and BoardgameQA, surpassing previous models that are trained on the full training set. Specifically, SatLM elevates the SoTA from 30.9% (Zhong et al., 2022) to 37.4% on LSAT and from 73.9% (Kazemi et al., 2023)) to 80.7% on BoardgameQA.



Q: Sara and Joe have a combined height of 120 inches. Joe is 6 inches more than double Sara's height. How tall is Joe?

```
sara_height = Variable()
joe_height = Variable()
total_height = 120
joe_height = 2 * sara_height + 6
total_height = sara_height + joe_height
solve(joe_height)
```

```
# We know total_height = sara_height + joe_height, and joe_height = 2 *
sara_height + 6.
# So total_height = sara_height + 2 * sara_height + 6 = 3 * sara_height + 6
# Therefore, sara_height = (total_height - 6) / 3 = (120 - 6) / 3 = 114 / 3 = 38
# joe_height = 2 * sara_height + 6 = 2 * 38 + 6 = 76 + 6 = 82
# The answer is 82
```

```
# The answer is 82
```

Figure 6.3: A variant of our approach which replaces the SAT solver with a "CoT solver" that takes the SAT problem as input and solves it in natural language.

In the regex synthesis domain, with greedy decoding, directly translating natural language descriptions to regexes (PROGLM) achieves 37.1%, whereas using declarative prompting achieves 44.0%. With self-consistency, we surpass the previous SoTA performance of 55.6% (Ye et al., 2021a).

6.4.3 Impact of SAT Solver & Declarative Prompting

Table 6.2: The performance of variants of our approach that use CoT Solver or No Solver. Using declarative prompting with CoT solver is more effective than imperative CoT prompting.

	GSM-SYS	GSM	CLUTRR
STANDARD	21.0	22.2	41.2
COT	46.5	62.7	40.8
PAL	43.4	72.8	58.9
SAT$_{\text{SYMSOLVER}}$	69.4	71.7	68.3
SAT$_{\text{COTSOLVER}}$	54.5	63.2	48.9
SAT$_{\text{NOSOLVER}}$	26.6	23.7	40.7

We conduct analysis to isolate the effectiveness of the two key components, the SAT solver and declarative prompting. Specifically, we test a variant of our approach that still uses declarative prompting but then solves the equations in natural language with CoT rather than using the symbolic solver (see Figure 6.3). Essentially, the LLM itself carries

Table 6.3: Fraction of planning errors (incorrect reasoning chains) and execution errors (numeric errors) made by CoTSolver.

	GSM-Sys	GSM	Clutrr
Plan Err	72.5	42.5	47.5
Exec Err	27.5	57.5	52.5

Table 6.4: Log likelihood (unnormalized / normalized) of the generated sequences (with greedy decoding) of ProgLM and SatLM on three datasets. Better log likelihood indicates higher LLM confidence in the parsing stage.

	GSM-Sys	GSM	Clutrr
PAL	$-9.5/-6.9_{10^{-2}}$	$\mathbf{-9.2/-6.0_{10^{-2}}}$	$-3.1/-8.5_{10^{-3}}$
SAT	$\mathbf{-8.5/-5.9_{10^{-2}}}$	$-9.7/-6.2_{10^{-2}}$	$\mathbf{-2.0/-7.9_{10^{-3}}}$

out planning and execution. This experiment helps isolate the benefits of the solver, which will compute an answer without making any mistakes, from the benefits of the declarative formulation. We also compare to prompting LLMs to directly give the answer (NoSolver).

Impact of Symbolic Solver As shown in Table 6.2, completely ablating the solver and directly predicting the answer (Sat$_{\text{NoSolver}}$) only yields performance that is on par with Standard. Interestingly, Sat$_{\text{CoTSolver}}$ can solve more SAT problems than NoSolver. This partially reflects the effectiveness of CoT and partially reflects the fact that many dataset instances require relatively simple planning and execution, allowing pure forward reasoning to solve them. However, using a symbolic solver (Sat$_{\text{SymSolver}}$), which guarantees correct planning and execution, leads to further improvements.

We manually analyzed 40 cases where the symbolic solver yields the correct answer but Sat$_{\text{CoTSolver}}$ fails to solve them. We categorized the errors as planning errors, where the reasoning chains are incorrect, and execution errors, where the reasoning chains are correct but computations are incorrect. Table 6.3 shows that most errors by Sat$_{\text{CoTSolver}}$ are planning errors, especially on GSM-Sys which requires solving complex system of equations.

Impact of Declarative Prompting Table 6.2 also shows that decoupling parsing and planning/solving is still useful, even when not using a symbolic solver: SAT$_{\text{CoTSolver}}$ outperforms CoT by 7.9%, and 8.1% on GSM-SYS and CLUTRR, respectively. We note that SAT$_{\text{CoTSolver}}$ can be viewed as a two-stage CoT prompting strategy, with a prompt showing that the first step is to formulate declaratively, then the next step is to solve.

We hypothesize that parsing a question into declarative formulas is more straightforward than parsing it into an imperative solving procedure. To evaluate this hypothesis, we use log likelihood of the generated tokens to assess how straightforward the translation is, as higher log-likelihood typically indicates the outputs are more fluent to LLMs, a connection demonstrated in Chapter 5 (as well as in recent literature (Gonen et al., 2022)). We show both unnormalized (total) and normalized log likelihood in Table 6.4. On GSM-SYS and CLUTRR where SATLM outperforms PROGLM, its generated outputs are also associated with higher likelihood.

6.4.4 Advantages of SAT in Selective Prediction

A SAT solver may not always return an answer, particularly if there are parsing errors from the question. We show that this is an advantage of SATLM: these errors allow us to abstain from making likely incorrect predictions.

Table 6.5 shows the fraction of correct predictions and incorrect predictions when the program or SAT solver successfully returns an answer as well as the fraction of different types of feedback signals. We report the fraction of questions *answered* as well as *selective accuracy*, defined by the fraction of overall accuracy (% of correct answers) normalized by coverage (% of answered problems). SATLM makes fewer predictions on all three datasets compared to PROGLM, as it can trigger both UNSAT and AMBIG errors. However, SATLM's selective accuracy is consistently better than PROGLM's, especially on GSM-SYS (77% vs 45%). As a result, SATLM's overall performance is significantly better than PROGLM on GSM-SYS and CLUTRR, even when making fewer predictions.

We note that on GSM, SATLM has slightly lower coverage but higher selective

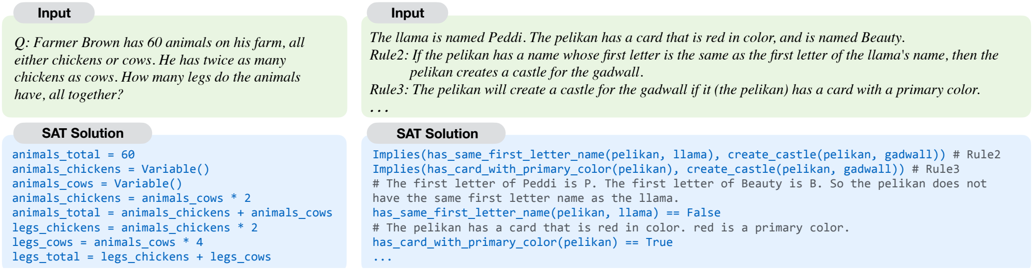
Table 6.5: Analysis of accuracy and execution status of SATLM and PROGLM. We present the fraction of tasks solved correctly or incorrectly in GSM-SYS, GSM, and CLUTRR, along with the breakdown of feedback from the solver. SATLM generally makes fewer predictions than PROGLM (ANSWERED), but more frequently makes correct predictions when it returns an answer (SELECTIVE ACC) *and* gives a higher absolute number of correct predictions on GSM-SYS and CLUTRR.

	GSM-SYS		GSM		CLUTRR	
	PROGLM	SATLM	PROGLM	SATLM	PROGLM	SATLM
CORRECT	43.3	69.4	72.7	71.8	58.9	68.3
INCORRECT	52.5	20.6	25.7	21.2	21.0	7.7
ERROR	4.2	2.6	1.6	2.1	20.1	3.5
UNSAT	–	2.4	–	1.5	–	15.5
AMBIG	–	5.0	–	3.4	–	5.0
ANSWERED	95.8	90.0	98.4	93.0	79.9	76.0
SELECTIVE ACC	45.2	**77.1**	73.8	**77.2**	73.7	**89.9**

accuracy compared to PROGLM. This explains why SATLM lags behind PROGLM with greedy decoding but outperforms PROGLM with self-consistency decoding (Table 6.1). By drawing multiple samples, SATLM can increase its coverage and achieve higher accuracy than PROGLM since its predictions are more accurate.

6.4.5 Analysis

LLMs Can Perform Commonsense Reasoning While Parsing There are many problems that do not state premises or constraints in a completely explicit way. Figure 6.4) shows two examples where commonsense inferences are required during parsing. For example, on the left, the model must recognize that *animals* refers to the chickens and cows collectively. Similarly, knowing that red is a primary color is needed to successfully apply rules on BOARDGAMEQA (right). We observe from the outputs in both cases that LLMs are capable of implicitly performing commonsense reasoning and produce correct logical formulas in the parsing step. As shown in Table 6.1, SATLM exhibits strong performance on BOARDGAMEQA, a dataset which requires this implicit background knowledge.



Input

Q: Farmer Brown has 60 animals on his farm, all either chickens or cows. He has twice as many chickens as cows. How many legs do the animals have, all together?

SAT Solution

```
animals_total = 60
animals_chickens = Variable()
animals_cows = Variable()
animals_chickens = animals_cows * 2
animals_total = animals_chickens + animals_cows
legs_chickens = animals_chickens * 2
legs_cows = animals_cows * 4
legs_total = legs_chickens + legs_cows
```

Input

The llama is named Peddi. The pelikan has a card that is red in color, and is named Beauty.
Rule2: If the pelikan has a name whose first letter is the same as the first letter of the llama's name, then the pelikan creates a castle for the gadwall.
Rule3: The pelikan will create a castle for the gadwall if it (the pelikan) has a card with a primary color.
. . .

SAT Solution

```
Implies(has_same_first_letter_name(pelikan, llama), create_castle(pelikan, gadwall)) # Rule2
Implies(has_card_with_primary_color(pelikan), create_castle(pelikan, gadwall)) # Rule3
# The first letter of Peddi is P. The first letter of Beauty is B. So the pelikan does not
have the same first letter name as the llama.
has_same_first_letter_name(pelikan, llama) == False
# The pelikan has a card that is red in color. red is a primary color.
has_card_with_primary_color(pelikan) == True
...
```

Figure 6.4: Examples outputs from GSM (left) and BOARDGAMEQA (right) show that LLMs can perform commonsense reasoning while parsing.

Table 6.6: Results on `gpt-3.5-turbo`, `text- davinci-003`, and `code-davinci-001`. The effectiveness of SATLM can generalize across LLMs.

	GSM-SYS	GSM	LSAT	CLUTRR	PROOF
	gpt-3.5-turbo (greedy decoding)				
COT	44.8	74.4	23.9	41.2	82.3
PROGLM	51.2	**77.9**	–	45.9	76.4
SATLM	**63.4**	76.4	**30.0**	**50.6**	**96.4**
	text-davinci-003 (greedy decoding)				
COT	42.8	62.5	21.7	34.5	83.5
PROGLM	40.4	**71.7**	–	41.2	83.7
SATLM	**63.6**	70.3	**30.4**	**58.2**	**99.7**
	code-davinci-001 (greedy decoding)				
PROGLM	15.5	**35.6**	–	22.2	63.8
SATLM	**16.5**	34.2	19.6	**30.2**	**86.6**

Results Across Different Language Models In addition to the main LLM used in our work, `code-davinci-002`, we further test whether SATLM can generalize to other LLMs. We choose `gpt-3.5-turbo` (0613 version), `text-davinci-003`, and `code-davinci-001`. `gpt-3.5-turbo` is optimized for chat. `text-davinci-003` is an LLM pretrained on NL, and tuned to align with human feedback (Ouyang et al., 2022). `code-davinci-001` is also an LLM pretrained on code, but less capable compared to `002`. As shown in Table 6.6, SATLM is better than PROGLM on the arithmetic reasoning and logical reasoning datasets except for GSM across these three LLMs. The trend is congruent with the results on `code-davinci-002` (Table 6.1), which suggests the approach's general applicability across different LLMs, regardless of their varying capabilities.

Table 6.7: The performance of PROGLM and SATLM with varying exemplar sets. SATLM consistently outperforms PROGLM on GSM-SYS and CLUTRR.

		GSM-SYS	GSM	CLUTRR
Set1	PROG	43.4	**72.7**	58.9
	SAT	**69.4**	71.8	**68.3**
Set2	PROG	41.4	**72.5**	59.0
	SAT	**71.8**	71.3	**67.9**
Set3	PROG	37.1	**70.3**	57.2
	SAT	**66.7**	70.0	**68.0**

Sensitivity to Different Exemplar Sets We test whether the advantages of SATLM is sensitive to different sets of exemplars. We experiment with 3 sets of exemplars on `code-davinci-002`. As shown in Table 6.7, SATLM consistently outperforms PROGLM by a large margin on GSM-SYS and CLUTRR, and achieves comparable performance on GSM. The results suggest the effectiveness of our approach is insensitive to varying the choice of exemplars.

6.5 Related Work

Reasoning with LLMs Our work is built on top of few-shot prompting (Brown et al., 2020), which has proven effective on a wide range of tasks (Wei et al., 2022b; Liu et al., 2023b; Gehrmann et al., 2021; Reif et al., 2022; Wei et al., 2022a; Sanh et al., 2022). In particular, we focus on improving LLMs on reasoning tasks, which are challenging for language models even with recent developments (Marcus, 2020; Garcez and Lamb, 2023). Various techniques have been proposed for improving reasoning abilities (Nye et al., 2021; Zhou et al., 2022a; Kojima et al., 2022; Khot et al., 2022; Fu et al., 2022; Wang et al., 2022a; Li et al., 2022a; Lyu et al., 2023). They largely follow a chain-of-thought (Wei et al., 2022c) or scratchpad (Nye et al., 2021) paradigm. Among them, our work is the most related to the line of work that generates imperative programs to be executed by a symbolic executor, such as a Python interpreter (Gao et al., 2023; Chen et al., 2022b) or domain-specific executors (Lyu et al., 2023). In this work, we propose a different paradigm that parses NL problems into declarative SAT problems and offloads the solving procedure to a SAT solver.

Tool-use Previous work has also explored equipping LLMs with other tools, including search engines (Yu et al., 2023; Schick et al., 2023), calculators (Cobbe et al., 2021; Chowdhery et al., 2022), or other domain-specific special modules (Schick et al., 2023; Demeter and Downey, 2020). A line of work focuses on using program-related tools such as program executors (Poesia et al., 2022), program analysis tools (Jain et al., 2022), and synthesis tools (Rahmani et al., 2021) to enhance the quality of the generated code. Our works further explores improving LLMs with SAT solvers.

Concurrent work explores the intersection of LLMs and planning, parsing planning problems into PDDL descriptions and leveraging a classical planner to produce the plan (Liu et al., 2023a). Our work differs in that we use the SAT formulation to solve general reasoning tasks, including arithmetic reasoning and logical reasoning, which cannot be specified in PDDL. Also concurrently, He-Yueya et al. (2023) combine LLMs and symbolic solvers for solving math problems. However, this work *only* focus on arithmetic reasoning tasks and employs a math-specific symbolic solver (PySym). Our work takes a more general approach by formulating the problem within the scope of first-order logic and therefore is domain-agnostic.

Semantic Parsing Our approach leverages LLMs to parse NL problems into SMT specifications, which is a form of semantic parsing (parsing NL descriptions into logical forms). Semantic parsing is a long-standing problem (Woods, 1973; Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Berant et al., 2013; Dong and Lapata, 2016). There have been various techniques proposed to translate natural language to "if-this-then-that" statements (Quirk et al., 2015), SQL queries (Iyer et al., 2017), bash commands (Lin et al., 2018), regular-expressions (Kushman and Barzilay, 2013; Ye et al., 2020c), and more. In particular, our method is closely aligned with previous efforts to parse natural language descriptions into Prolog (or Prolog-like) logical forms (Pereira and Warren, 1980; Dowding et al., 1993;

Tang and Mooney, 2001), which is also a type of declarative logical specification. These systems primarily focus on using Prolog to handle NL queries for interacting with domain-specific knowledge bases, whereas our approach aims to enhance the reasoning capabilities of LLMs. We employ code-style SMT specifications and leverage general-purpose LLMs to address a broad range of reasoning tasks. Additionally, while most prior work in semantic parsing focuses on single-sentence queries, our method is capable of handling complex problems that involve multiple constraints (e.g., each example for LSAT involves more than 8 constraints on average).

6.6 Conclusion & Limitations

We have presented a framework for satisfiability-aided language models, casting a wide range of reasoning tasks into SAT problems under a unified formulation. We use an LLM to parse an NL query into a declarative specification and leverages a SAT solver to derive the final answer. Evaluation results on 8 datasets spanning 4 tasks across several LLMs demonstrate the effectiveness of our approach over program-aided language models.

Limitations Our framework parses an NL problems into a set of declarative formulas. The NL description of some problems may already be more compatible with an imperative solving procedure, and our approach is likely to be less effective in these cases (e.g., SATLM slightly lags PROGLM on GSM). Future research can explore an integration or ensemble of these two prompting styles for more flexible reasoning.

SATLM heavily relies on the SAT solver and inherits some limitations of the SAT solver itself, such as computational cost when dealing with complex formulas involving quantifiers or nonlinear arithmetic. Moreover, SAT solvers can be limited by the expressiveness of the underlying theory, as not all theories can be easily encoded in first-order logic. Nevertheless, the wide range of tasks that we can instantiate our SATLM framework on shows its general applicability.

Our current approach parses a problem into a SAT specification, runs the solver, and returns the answer in a one-round fashion. One can imagine that unsatisfiable formulas or ambiguous formulas could be improved by re-prompting the model to improve the specification based on the exception signals, as explored in concurrent work for other problems (Paul et al., 2023; Madaan et al., 2023; Chen et al., 2023). We believe this is an exciting direction for future work.

Chapter 7: Conclusion and Future Work

In this dissertation, our goal is to enable language models perform complex reasoning reliably. We fullfill this goal by building frameworks for steering textual reasoning with explanations.

Intervening on Predictions Post-Hoc In Chapters 2 - 4, we have established a framework that leverages explanations for improving model predictions via post-hoc intervention. We start from a case study that connects explanations and QA model behavior in Chapter 2. We present a new methodology using explanations to understand model behavior on realistic counterfactuals. We show explanations can indeed be connected to model behavior, and therefore we can compare explanations to understand which ones truly give us actionable insights about what our models are doing.

Our findings mentioned above motivate us to explore, in Chapter 3, whether model attributions can be useful for calibrating black box models. The answer is yes. We build a framework that trains a verifer to assessing model robustness based on features describing the reasoning process. Using the features automatically extracted from attributions for BERT-based models, we improve model generalization performance on new domains and tasks. In addition, it exhibits promising generalization performance in cross-domain generalization and Selective answering.

In Chapter 4, we shift our focus to studying the capabilities of LLMs in using explanations in in-context learning for textual reasoning, given their success in various reasoning tasks. Through our experiments with four LLMs and on two QA datasets and an NLI dataset, we find that simply including explanations in the prompt does not always improve the performance of in-context learning. Our manual analysis demonstrates that LLMs tend to generate nonfactual explanations when making wrong predictions. Nevertheless, nonfactual explanations can be a useful leverage to assess the correctness of the predictions. We use the general framework developed in Chapter 3, building lightweight calibrators that uses features approximating the factuality of explanations from LLMs. Our framework successfully improve LLMs' in-context learning performance across all three datasets.

Teaching LLMs to Reason with Explanations In addition post-hoc intervention, we also explore how to use explanations as training supervision for teaching LMs to reason. In Chapter 5 - 6, we particularly focus on the question of how to formalize more effective explanations for LLMs, as using different explanations can lead to substantially varied performance on downstream tasks.

In Chapter 5, we present an approach that can search for better-performing explanations for in-context learning starting from a set of seed explanations. Our approach first proposes promising candidate combinations of alternative explanations generated using LLMs, then finds explanation combinations using proxy metrics before using a silver-labeled validation set to select the best candidate. Through preliminary investigation, we highlight the substantial variance in the performance of different sets of explanations. Further evaluation results suggest that using our framework can find substantially better explanations measured by prompting performance compared to seed explanations.

Our optimization technique can substantially improve downstream performance by finding the best "paraphrases" of the original explanations. In Chapter 6, we study how a fundamental characteristic of the explanations, whether they are imperative or declarative, impacts the effectiveness of explanations. We present a framework for satisfiability-aided language models. Our framework uses declarative formal specifications as the explanations and equips LLMs with SMT solvers, which amends the limited planning capabilities of LLMs. We first cast a wide range of reasoning tasks into SAT problems under a unified formulation. Next, we use an LLM to parse an NL query into a declarative specification and leverages a SAT solver to derive the final answer. Evaluation results on multiple datasets spanning arithmetic reasoning, logical reasoning, symbolic reasoning, and regex synthesis, across several LLMs demonstrate the effectiveness of our approach over baselines that use imperative explanations; our framework scales LLMs to much more complex reasoning problems such as LSAT problems exhibiting extensively long reasoning steps.

7.1 Future Directions

Our ultimate goal is to augment human capabilities with LMs in various tasks demanding deep reasoning (such as data analytics and programming), surpassing the efficacy and efficiency achievable by humans alone. We believe that leveraging explanations is crucial to enable robust reasoning skills and effective human-LM collaboration, which are essential for realizing this goal. Finally, we outline future directions to further advance our research agenda.

Effective human-LM collaboration with explanations as the vehicle Despite the variety of explanation forms and generation techniques available, extensive research suggests that explanations have only achieved limited success in aiding humans across many tasks (Hase and Bansal, 2020; Wang and Yin, 2021; Joshi et al., 2023). To this end, we are interested in developing a more effective protocol for human-LM collaboration, where LMs can take initiative to seek explanations towards collaboratively solving a problem. Specifically, LMs can actively express their uncertainties and request clarifications on sub-tasks or data instances where their confidence is low. In turn, humans can inspect their explanations on these instances as well and provide targeted feedback to guide LM behavior. This protocol raises intriguing questions regarding the most effective forms of explanations for LMs (such as case-based or contrastive explanations (Jacovi et al., 2021)) and feedback (like natural language instructions or preferences regarding explanations). This protocol also raises questions about how to enhance LMs to provide better explanations for humans. Future research can potentially work on optimizing LM-generated explanations based on their utility to human users.

Combining NL explanations and formal explanations for flexible and robust reasoning While utilizing formal specifications to teach LMs guarantees soundness in reasoning, there are many reasoning tasks that involve both “hard” constraints as well as rules that are tricky to articulate solely through formal specifications. For instance, in legal reasoning, certain prerequisites must be met for a verdict of guilt, yet whether a suspect in a case fulfills these criteria can be debatable and difficult to verify with formal specifications alone. Furthermore, some problems might require a blend of reasoning types, some that are suitable for solvers like deductive reasoning, and others less so, such as defeasible reasoning and reasoning by analogy. A system that can make use of both hard formal specifications (like SMT formulas) as well NL statements (NL proposition based on commonsense) would take benefit of both the reliability of symbolic systems as well as the flexibility of LM’s capabilities in NL reasoning. Future research can work on designing neurosymbolic specifications and solvers for complex reasoning tasks.

Building resources towards complex reasoning in real-world applications LMs have significantly advanced in their reasoning capabilities, demonstrating great potential for a wide range of applications. However, there is a notable gap in resources for benchmarking and enhancing LMs’ reasoning abilities in a manner that aligns with actual user needs in real-world scenarios. Many existing resources for language reasoning are synthetic in nature, both in terms of language used and the goals of the tasks. Some of our previous work has involved compiling datasets that test reasoning with natural narrative text (Sprague et al., 2023) or real user queries (Ye et al., 2020a). Moving forward, our aim is to develop resources that facilitate the application of LLMs in real-world settings where language-based reasoning is crucial. One area of focus is data analysis tasks, which require in-depth examination of data to extract meaningful insights. For instance, how can we enable LMs to analyze sales data, pinpointing key factors and customer segments that could drive revenue growth? This task demands both data-driven reasoning and commonsense reasoning. Potential future work can be dedicated to establishing datasets and platforms that support research in this direction.

Works Cited

Julius Adebayo, Michael Muelly, Harold Abelson, and Been Kim. Post hoc explanations may be ineffective for detecting unknown spurious correlation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xNOVfCCvDpM.

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. Explanations for CommonsenseQA: New Dataset and Models. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2021.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Jasmijn Bastings, Sebastian Ebert, Polina Zablotskaia, Anders Sandholm, and Katja Filippova. ”Will You Find These Shortcuts?” A Protocol for Evaluating the Faithfulness of Input Salience Methods for Text Classification. In *arXiv*, 2021. doi: 10.48550/ARXIV.2111.07367. URL https://arxiv.org/abs/2111.07367.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

Emily M. Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2020.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL `https://doi.org/10.1145/3442188.3445922`.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D13-1160`.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, 2017.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-SNLI: Natural Language Inference with Natural Language Explanations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Arjun Chandrasekaran, Viraj Prabhu, Deshraj Yadav, Prithvijit Chattopadhyay, and Devi Parikh. Do explanations make VQA models more predictable to a human? In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, 2005.

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. Generating hierarchical explanations on text classification via feature interaction detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Howard Chen, Jacqueline He, Karthik Narasimhan, and Danqi Chen. Can rationalization improve robustness? In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2022a.

Jifan Chen and Greg Durrett. Understanding dataset design choices for multi-hop reasoning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

Jifan Chen, Eunsol Choi, and Greg Durrett. Can NLI models verify QA systems' predictions? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura 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, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. 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. *ArXiv*, abs/2107.03374, 2021b.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022b.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Baindoor Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas

Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways. *ArXiv*, abs/2204.02311, 2022.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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)*, 2019.

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. *arXiv preprint arXiv:2110.14168*, 2021.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018.

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=3Pf3Wg6o-A4`.

Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, jul 1960. ISSN 0004-5411. doi: 10.1145/321033.321034. URL `https://doi.org/10.1145/321033.321034`.

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)*, 2020.

Leonardo De Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, page 337–340, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3540787992.

David Demeter and Doug Downey. Just add functions: A neural-symbolic language model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7634–7642, Apr. 2020. doi: 10.1609/aaai.v34i05.6264. URL https://ojs.aaai.org/index.php/AAAI/article/view/6264.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2020.

Shizhe Diao, Xuechun Li, Yong Lin, Zhichao Huang, Xiao Zhou, and Tong Zhang. Black-box prompt learning for pre-trained language models. *ArXiv*, abs/2201.08531, 2022. URL https://api.semanticscholar.org/CorpusID:246210164.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Li Dong and Mirella Lapata. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1004. URL https://www.aclweb.org/anthology/P16-1004.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

John Dowding, Jean Mark Gawron, Doug Appelt, John Bear, Lynn Cherny, Robert Moore, and Douglas Moran. GEMINI: A natural language system for spoken-language understanding. In *31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, USA, June 1993. Association for Computational Linguistics.

Dheeru Dua, Sameer Singh, and Matt Gardner. Benefits of intermediate annotations in reading comprehension. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2020.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Fkckkr3ya8.

Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010. URL http://jmlr.org/papers/v11/el-yaniv10a.html.

Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott M Lundberg, and Su-In Lee. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature machine intelligence*, 3(7):620–631, 2021.

Aarohi Srivastava et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *ArXiv*, abs/2206.04615, 2022.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of 2nd Machine Reading for Reading Comprehension (MRQA) Workshop at EMNLP*, 2019.

Victoria Fossum and Kevin Knight. Combining constituent parsers. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, 2009.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

Artur d’Avila Garcez and Luis C Lamb. Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review*, pages 1–20, 2023.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

Siddhant Garg and Alessandro Moschitti. Will this Question be Answered? Question Filtering via Answer Model Distillation for Efficient Question Answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Anuoluwapo Aremu, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh Dhole, Wanyu Du, Esin Durmus, Ondřej Dušek, Chris Chinenye Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Mihir Kale, Dhruv Kumar, Faisal Ladhak, Aman Madaan, Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Andre Niyongabo Rubungo, Salomey Osei, Ankur Parikh, Laura Perez-Beltrachini, Niranjan Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc, Thibault Sellam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabezudo, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. The GEM benchmark: Natural language generation, its evaluation and metrics. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, pages 96–120, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.gem-1.10. URL https://aclanthology.org/2021.gem-1.10.

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, 2021a. doi: 10.1162/tacl_a_00370. URL https://aclanthology.org/2021.tacl-1.21.

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 (TACL)*, 2021b.

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. Demystifying prompts in language models via perplexity estimation. *arXiv preprint arXiv:2212.04037*, 2022.

Dirk Groeneveld, Tushar Khot, Mausam, and Ashish Sabharwal. A simple yet strong pipeline for HotpotQA. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Chaoyu Guan, Xiting Wang, Quanshi Zhang, Runjin Chen, Di He, and Xing Xie. Towards a deep and unified understanding of deep neural models in NLP. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. Multi-source domain adaptation for text classification via distancenet-bandits. In *Proceedings of the Annual Meeting of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher Ré. Training classifiers with natural language explanations. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2018.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Self-attention attribution: Interpreting information interactions inside transformer. In *Proceedings of the Annual Meeting of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

David Harbecke. Explaining natural language processing classifiers with occlusion and language modeling. *arXiv preprint arXiv:2101.11889*, 2021.

Peter Hase and Mohit Bansal. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Joy He-Yueya, Gabriel Poesia, Rose E. Wang, and Noah D. Goodman. Solving math word problems by combining language models with symbolic solvers. *ArXiv*, abs/2304.09102, 2023.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating Visual Explanations. In *European Conference on Computer Vision (ECCV)*, 2016.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

J. Hewitt and P. Liang. Designing and interpreting probes with control tasks. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. *arXiv preprint arXiv:2108.02035*, 2021.

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

Yukun Huang, Yixin Liu, Raghuveer Thirukovalluru, Arman Cohan, and Bhuwan Dhingra. Calibrating long-form generations from large language models. *arXiv preprint arXiv:2402.06544*, 2024.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. Learning a neural semantic parser from user feedback. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

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*, 2020.

Alon Jacovi and Yoav Goldberg. Aligning faithful interpretations with their social attribution. *Transactions of the Association for Computational Linguistics (TACL)*, 9: 294–310, 2021. doi: 10.1162/tacl_a_00367.

Alon Jacovi, Swabha Swayamdipta, Shauli Ravfogel, Yanai Elazar, Yejin Choi, and Yoav Goldberg. Contrastive explanations for model interpretability. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1597–1611, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.120.

Naman Jain, Skanda Vaidyanath, Arun Iyer, Nagarajan Natarajan, Suresh Parthasarathy, Sriram Rajamani, and Rahul Sharma. Jigsaw: Large language models meet program synthesis. *ICSE*, May 2022.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2017.

Yichen Jiang and Mohit Bansal. Avoiding reasoning shortcuts: Adversarial evaluation, training, and model development for multi-hop QA. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Xisen Jin, Zhongyu Wei, Junyi Du, Xiangyang Xue, and Xiang Ren. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. In *International Conference on Learning Representations*, 2020.

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. 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 7103–7128, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.392.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

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 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Amita Kamath, Robin Jia, and Percy Liang. Selective question answering under domain shift. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Sklgs0NFvr`.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. BoardgameQA: A Dataset for Natural Language Reasoning with Contradictory Information. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

Nate Kushman and Regina Barzilay. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2013.

Matthew Lamm, Jennimaria Palomaki, Chris Alberti, Daniel Andor, Eunsol Choi, Livio Baldini Soares, and Michael Collins. QED: A framework and dataset for explanations in question answering. *Transactions of the Association for Computational Linguistics*, 9:790–806, 2021. doi: 10.1162/tacl_a_00398. URL `https://aclanthology.org/2021.tacl-1.48`.

Andrew K Lampinen, Ishita Dasgupta, Stephanie CY 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? In *Findings of the Conference on Empirical Methods in Natural Language Processing (Findings of EMNLP)*, 2022.

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, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1011. URL `https://www.aclweb.org/anthology/D16-1011`.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. *ArXiv*, abs/2206.14858, 2022.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, et al. Explanations from large language models make small reasoners better. *arXiv preprint arXiv:2210.06726*, 2022a.

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. *arXiv preprint arXiv:2206.02336*, 2022b.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. NL2Bash: A Corpus and Semantic Parser for Natural Language Interface to the Linux Operating System. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation LREC 2018, Miyazaki (Japan), 7-12 May, 2018.*, 2018.

Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+ P: Empowering Large Language Models with Optimal Planning Proficiency. *arXiv preprint arXiv:2304.11477*, 2023a.

Frederick Liu and Besim Avci. Incorporating priors with feature attribution on text classification. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2019.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.deelio-1.10. URL `https://aclanthology.org/2022.deelio-1.10`.

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 Comput. Surv.*, 55(9), jan 2023b. ISSN 0360-0300. doi: 10.1145/3560815. URL `https://doi.org/10.1145/3560815`.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, May 2022.

Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*, 2023.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. Domain adaptation with BERT-based domain classification and data selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, 2019.

Aman Madaan and Amir Yazdanbakhsh. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*, 2022.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Ana Marasović, Iz Beltagy, Doug Downey, and Matthew E. Peters. Few-shot self-rationalization with natural language prompts. In *Findings of the North American Chapter of the Association for Computational Linguistics (NAACL Findings)*, 2022.

Gary Marcus. The next decade in AI: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*, 2020.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL `https://aclanthology.org/J93-2004`.

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*, 2019.

Sabrina J. Mielke, Arthur D. Szlam, Y.-Lan Boureau, and Emily Dinan. Linguistic calibration through metacognition: aligning dialogue agent responses with expected correctness. *ArXiv*, abs/2012.14983, 2020.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. Compositional questions do not necessitate multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. Reframing instructional prompts to GPTk's language. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

Jesse Mu and Jacob Andreas. Compositional explanations of neurons. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Dong Nguyen. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-tau Yih, Sida I Wang, and Xi Victoria Lin. LEVER: Learning to Verify Language-to-Code Generation with Execution. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

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. *ArXiv*, abs/2112.00114, 2021.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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*, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL `https://aclanthology.org/2021.naacl-main.168`.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*, 2023.

Fernando C.N. Pereira and David H.D. Warren. Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13(3):231–278, 1980. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(80)90003-X.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. In *Proceedings of the International Conference on Learning Representations Workshop (ICLR Workshop)*, 2017.

John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Gregory Plumb, Maruan Al-Shedivat, Ángel Alexander Cabrera, Adam Perer, Eric Xing, and Ameet Talwalkar. Regularizing black-box models for improved interpretability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Gabriel Poesia, Alex Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=KmtVD97J43e`.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models. In *arXiv*, 2022.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *ArXiv*, abs/2210.03350, 2022.

Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark, and Yejin Choi. Counterfactual story reasoning and generation. 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), 2019.

Chris Quirk, Raymond Mooney, and Michel Galley. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 878–888, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1085. URL `https://www.aclweb.org/anthology/P15-1085`.

Kia Rahmani, Mohammad Raza, Sumit Gulwani, Vu Le, Daniel Morris, Arjun Radhakrishna, Gustavo Soares, and Ashish Tiwari. Multi-modal program inference: A marriage of pre-trained language models and component-based synthesis. *Proc. ACM Program. Lang.*, 5(OOPSLA), oct 2021. doi: 10.1145/3485535. URL `https://doi.org/10.1145/3485535`.

Nazneen Fatema Rajani and Raymond Mooney. Stacking with auxiliary features for visual question answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana, 2018.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain Yourself! Leveraging Language Models for Commonsense Reasoning. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages

2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://www.aclweb.org/anthology/D16-1264.

Alan Ramponi and Barbara Plank. Neural unsupervised domain adaptation in NLP—A survey. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2020.

Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. A recipe for arbitrary text style transfer with large language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 837–848, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.94. URL https://aclanthology.org/2022.acl-short.94.

Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Henghui Zhu, Rui Dong, Deguang Kong, Juliette Burger, Anjelica Ramos, zhiheng huang, William Yang Wang, George Karypis, Bing Xiang, and Dan Roth. STREET: A multi-task structured reasoning and explanation benchmark. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1C_kSW1-k0.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, volume 18, pages 1527–1535, 2018.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Laura Rieger, Chandan Singh, William Murdoch, and Bin Yu. Interpretations are useful: Penalizing explanations to align neural networks with prior knowledge. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2022.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1, 2019.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=9Vrb9D0WI4`.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=qFVVBzXxR2V`.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv*, 2023. doi: 10.48550/ARXIV.2302.04761. URL `https://arxiv.org/abs/2302.04761`.

Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. Constrained language models yield few-shot semantic parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6034`.

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)*, November 2019.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

Joe Stacey, Yonatan Belinkov, and Marek Rei. Supervising model attention with human explanations for robust natural language inference. In *Proceedings of the Annual Meeting of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2022.

Julia Strout, Ye Zhang, and Raymond Mooney. Do Human Rationales Improve Machine Explanations? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–62, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4807. URL `https://www.aclweb.org/anthology/W19-4807`.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3): 647–665, 2014.

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 Annual Meeting of the Association for Computation Linguistics (ACL)*, 2020.

Qiushi Sun, Chengcheng Han, Nuo Chen, Renyu Zhu, Jing Gong, Xiang Lisa Li, and Ming Gao. Make prompt-based black-box tuning colorful: Boosting model generalization from three orthogonal perspectives. *ArXiv*, abs/2305.08088, 2023. URL `https://api.semanticscholar.org/CorpusID:258685392`.

Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuanjing Huang, and Xipeng Qiu. BBTv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, December 2022a.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. *arXiv preprint arXiv:2201.03514*, 2022b.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Oyvind Tafjord, 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 (ACL Findings)*, August 2021.

Alon Talmor and Jonathan Berant. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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)*, June 2019.

Lappoon R Tang and Raymond J Mooney. Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*, pages 466–477. Springer, 2001.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*, 2019.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Generating token-level explanations for natural language inference. 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 963–969, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1101. URL `https://www.aclweb.org/anthology/N19-1101`.

Michael Tsang, Sirisha Rambhatla, and Yan Liu. How does this interaction affect me? interpretable attribution for feature interactions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). *ArXiv*, abs/2206.10498, 2022.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Elena Voita and Ivan Titov. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

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)*, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

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. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2023.

Peifeng Wang, Aaron Chan, Filip Ilievski, Muhao Chen, and Xiang Ren. Pinto: Faithful language reasoning using prompt-generated rationales. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022a.

Xinru Wang and Ming Yin. Are explanations helpful? a comparative study of the effects of explanations in ai-assisted decision-making. In *Proceedings of the 26th International Conference on Intelligent User Interfaces*, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380171. doi: 10.1145/3397481.3450650.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Rationale-augmented ensembles in language models. *ArXiv*, abs/2207.00747, 2022b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022c.

Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022a. URL `https://openreview.net/forum?id=gEZrGCozdqR`.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022b. URL `https://openreview.net/forum?id=yzkSU5zdwD`. Survey Certification.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022c.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Sarah Wiegreffe 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)*, 2019.

Sarah Wiegreffe, Ana Marasović, and Noah A. Smith. Measuring association between labels and free-text rationales. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Sarah Wiegreffe, Jack Hessel, Swabha Swayamdipta, Mark Riedl, and Yejin Choi. Reframing Human-AI Collaboration for Generating Free-Text Explanations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

W. A. Woods. Progress in natural language understanding: an application to lunar geology. In *AFIPS National Computer Conference*, AFIPS '73, page 441–450, New York, NY, USA, 1973. Association for Computing Machinery. ISBN 9781450379168. doi: 10.1145/1499586.1499695. URL `https://doi.org/10.1145/1499586.1499695`.

Jialin Wu and Raymond Mooney. Faithful multimodal explanation for visual question answering. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Self-evaluation guided beam search for reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Huihan Yao, Ying Chen, Qinyuan Ye, Xisen Jin, and Xiang Ren. Refining language models with compositional explanations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Xi Ye and Greg Durrett. Can explanations be useful for calibrating black box models? In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2022a.

Xi Ye and Greg Durrett. The unreliability of explanations in few-shot prompting for textual reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b.

Xi Ye and Greg Durrett. Explanation selection using unlabeled data for chain-of-thought prompting. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Benchmarking multimodal regex synthesis with complex structures. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2020a.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Benchmarking multimodal regex synthesis with complex structures. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2020b.

Xi Ye, Qiaochu Chen, Xinyu Wang, Isil Dillig, and Greg Durrett. Sketch-driven regular expression generation from natural language and examples. In *Transactions of the ACL*, 2020c.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Optimal neural program synthesis from multimodal specifications. In *Findings of the Association for Computational Linguistics: EMNLP (EMNLP Findings)*, 2021a.

Xi Ye, Rohan Nair, and Greg Durrett. Connecting attributions and qa model behavior on realistic counterfactuals. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021b.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: Satisfiability-aided language models using declarative prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a.

Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Ves Stoyanov, Greg Durrett, and Ramakanth Pasunuru. Complementary explanations for effective in-context learning. In *Findings of the Annual Meeting of the As- sociation for Computational Linguistics (ACL Findings)*, 2023b.

Chih-Kuan Yeh, Been Kim, Sercan Ö. Arik, C. Li, P. Ravikumar, and T. Pfister. On concept-based explanations in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Pengcheng Yin and Graham Neubig. Reranking for neural semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. Generate rather than retrieve: Large language models are strong context generators. In *International Conference for Learning Representation (ICLR)*, 2023.

Omar Zaidan, Jason Eisner, and Christine Piatko. Using "annotator rationales" to improve machine learning for text categorization. In *Proceedings of the Annual Meeting of the Association for Computation Linguistics (ACL)*, 2007.

Eric Zelikman, Yuhuai Wu, and Noah D Goodman. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the Annual Meeting of the Association for the Advancement of Artificial Intelligence (AAAI)*, AAAI'96, page 1050–1055. AAAI Press, 1996. ISBN 026251091X.

Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, page 658–666, Arlington, Virginia, USA, 2005. AUAI Press. ISBN 0974903914.

Shujian Zhang, Chengyue Gong, and Eunsol Choi. Knowing more about questions can help: Improving calibration in question answering. In *Findings of the Annual Conference of the Association for Computational Linguistics (ACL Findings)*, 2021.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open Pre-trained Transformer Language Models. *ArXiv*, abs/2205.01068, 2022a.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E Gonzalez. Tempera: Test-time prompting via reinforcement learning. *arXiv preprint arXiv:2211.11890*, 2022b.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*, 2023.

Theodore Zhao, Mu Wei, J Samuel Preston, and Hoifung Poon. Automatic calibration and error correction for large language models via pareto optimal self-supervision. *arXiv preprint arXiv:2306.16564*, 2023.

Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Yining Chen, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. Analytical reasoning of text. In *Findings of the Association for Computational Linguistics: NAACL (NAACL Findings)*, 2022.

Denny Zhou, Nathanael Scharli, 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. *ArXiv*, abs/2205.10625, 2022a.

Hattie Zhou, Azade Nova, H. Larochelle, Aaron C. Courville, Behnam Neyshabur, and Hanie Sedghi. Teaching algorithmic reasoning via in-context learning. *ArXiv*, abs/2211.09066, 2022b.

Yangqiaoyu Zhou and Chenhao Tan. Investigating the effect of natural language explanations on out-of-distribution generalization in few-shot NLI. In *Proceedings of the Workshop on Insights from Negative Results in NLP*, 2021.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022c.

Chiwei Zhu, Benfeng Xu, Quan Wang, Yongdong Zhang, and Zhendong Mao. On the calibration of large language models and alignment. *arXiv preprint arXiv:2311.13240*, 2023.

Vita

Xi Ye received the Bachelor of Engineering degree in Software Engineering from the School of Software at Tsinghua University in 2018. During his undergraduate studies, he worked as a research assistant in THUVis Lab under the supervision of Shixia Liu, focusing on data visualization and machine learning. After that, he joined the Department of Computer Science at The University of Texas at Austin as a graduate student, where he pursued his doctoral research under the supervision of Greg Durrett. His research is in the area of natural language processing, particularly in leveraging explanations to steer language models for complex textual reasoning tasks. He is also interested in semantic parsing and program synthesis.

Address: xiye@cs.utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.