
Xiao Liu¹, Da Yin², Chen Zhang¹, Yansong Feng¹,³ and Dongyan Zhao¹,⁴,⁵

¹Wangxuan Institute of Computer Technology, Peking University
²Computer Science Department, University of California, Los Angeles
³The MOE Key Laboratory of Computational Linguistics, Peking University
⁴Beijing Institute for General Artificial Intelligence
⁵State Key Laboratory of Media Convergence Production Technology and Systems
{lxlisa,zhangch,fengyansong,zhaody}@pku.edu.cn
da.yin@cs.ucla.edu

Abstract

Causal reasoning, the ability to identify cause-and-effect relationship, is crucial in human thinking. Although large language models (LLMs) succeed in many NLP tasks, it is still challenging for them to conduct complex causal reasoning like abductive reasoning and counterfactual reasoning. Given the fact that programming code may express causal relations more often and explicitly with conditional statements like if, we want to explore whether Code-LLMs acquire better causal reasoning abilities. Our experiments show that compared to text-only LLMs, Code-LLMs with code prompts are significantly better in causal reasoning. We further intervene on the prompts from different aspects, and discover that the programming structure is crucial in code prompt design, while Code-LLMs are robust towards format perturbations. Code and data are available at https://github.com/xxxiaol/magic-if.

1 Introduction

Human beings rely heavily on the capacity for causal reasoning (Sloman, 2005; Hagmayer et al., 2007). People understand the observed facts, predict future events, and speculate about what might have happened if things had been different with the help of their causal reasoning skills. For instance, when we go home and find a mess, we probably want to figure out why it happened. If we determine that a bird flew into the house, we might then consider whether the mess could have been avoided if we had closed the window.

Although large language models (LLMs) demonstrate great language understanding and generation abilities, it is still challenging for them to perform complex causal reasoning such as the example above. Powerful LLMs are able to understand single cause-and-effect relations (Brown et al., 2020; Wang et al., 2021), like a man losing his balance causes him to fell. However, when it comes to more complex causal structures involving multiple events and alternative branches (like close the window or not), LLMs perform much inferior to humans (Bhagavatula et al., 2019; Qin et al., 2019). In this paper, we consider two challenging causal reasoning tasks: abductive reasoning and counterfactual reasoning. Abductive reasoning requires models to generate a plausible reason for the ending while being consistent with the premise. Counterfactual reasoning asks what will occur in the counterfactual branch. Causal relationships between events in these tasks are shown in Figure 1.

A potential difficulty for LLMs to learn complex
causal structures is that they are rarely expressed explicitly in the text. News articles or narratives may contain multiple events with causal relationships, like an incident and a chain of consequences. However, these events are often written chronologically, and it is hard to extract the causal structure from the text without further annotation. Branches are expressed rarer in text, except for the multi-branching storytelling style (Nisi and Haahr, 2006).

On the other hand, causal relations are exhibited more commonly in code. Conditional statements like if direct the computer to execute certain commands, provided a condition is met. This explicitly demonstrates the causal relationship between the condition block and the execution block. Code can also express branching with else or switch statements, and the nesting feature enables code to describe more complex structures1.

This motivates us to utilize code models in natural language causal reasoning. Recently, large language models of code (Code-LLMs) are receiving increasing attention (Chen et al., 2021; Xu et al., 2022). They exhibit strong code generation performance, and their structural prediction abilities help complete structural natural language tasks like argument graph generation (Madaan et al., 2022) and event argument extraction (Wang et al., 2022b). Being pre-trained on code with abundant causal expressions, Code-LLMs may also have gained better causal reasoning abilities.

We conduct experiments on the unsupervised abductive reasoning and counterfactual reasoning tasks. To generate task outputs, we design code prompts like Figure 2 to clearly represent the causal structures of the tasks. Results show that Code-LLMs with code prompts perform much better than text-only LLMs and previous methods. To better understand why the code prompts are effective, we break down the prompts and analyze the influence of different aspects. We find that Code-LLMs are very sensitive to the programming structure (specifically, the conditional statements), while being robust towards format perturbations and programming language changes.

Our main contributions are as follows: 1) We design code prompts to tackle causal reasoning tasks, by leveraging conditional statements in code to represent causal structures. 2) We evaluate Code-LLMs with code prompts on the abductive reasoning and counterfactual reasoning tasks, and exhibit that code models with code prompts are better causal reasoners than text models. 3) We break down the code prompt in detail and find that the programming structure is crucial to the performance.

2 Modeling Causal Structure with Code

We convert the input of causal reasoning tasks into the form of code prompt for Code-LLMs to understand better. We expect the prompts to meet two requirements: 1) clearly represent the causal relationships between events, and 2) as most Code-LLMs only support generating at the end, the target output should appear at the end of the prompts. The first requirement is addressed with conditional statements. However, for the second, the target prediction is not always the last part of the conditional statements, e.g., in abductive reasoning we want to predict the hypothesis, which is the condition in the if structure. To address this, we uniformly use functions to represent events. As shown in Figure 2, the causal structure is described in the main function. All the event functions are listed afterwards.

---

1 Although causal expressions like if are also used in natural languages, representing complex causal structures in text is not as clear and structured as in code.
Table 1: Automatic evaluation results on two unsupervised causal reasoning tasks in the zero-shot setting. Numbers are in percentages (%).

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU</th>
<th>ROUGE</th>
<th>CIDEr</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELOREAN</td>
<td>1.6</td>
<td>19.1</td>
<td>7.9</td>
<td>41.7</td>
</tr>
<tr>
<td>COLD</td>
<td>1.8</td>
<td>19.5</td>
<td>10.7</td>
<td>42.7</td>
</tr>
<tr>
<td>DIFFUSION</td>
<td>7.1</td>
<td>28.3</td>
<td>30.7</td>
<td>42.7</td>
</tr>
<tr>
<td>DAVINCI002</td>
<td>4.9</td>
<td>27.0</td>
<td>26.6</td>
<td>56.8</td>
</tr>
<tr>
<td>DAVINCI003</td>
<td>4.6</td>
<td>25.8</td>
<td>10.7</td>
<td>57.1</td>
</tr>
<tr>
<td>CODEX</td>
<td>13.7</td>
<td>39.6</td>
<td>81.8</td>
<td>64.9</td>
</tr>
</tbody>
</table>

(a) Abductive reasoning.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU</th>
<th>ROUGE</th>
<th>CIDEr</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELOREAN</td>
<td>21.4</td>
<td>40.7</td>
<td>63.4</td>
<td></td>
</tr>
<tr>
<td>CGMHI</td>
<td>41.3</td>
<td>-</td>
<td>73.8</td>
<td></td>
</tr>
<tr>
<td>EDUCAT</td>
<td>44.1</td>
<td>-</td>
<td>74.1</td>
<td></td>
</tr>
<tr>
<td>DAVINCI002</td>
<td>49.0</td>
<td>54.7</td>
<td>73.0</td>
<td></td>
</tr>
<tr>
<td>DAVINCI003</td>
<td>30.6</td>
<td>45.2</td>
<td>69.4</td>
<td></td>
</tr>
<tr>
<td>CODEX</td>
<td>66.8</td>
<td>70.0</td>
<td>82.5</td>
<td></td>
</tr>
</tbody>
</table>

(b) Counterfactual reasoning.

Table 2: Human evaluation of comparing CODEX and DAVINCI002.

<table>
<thead>
<tr>
<th>Method</th>
<th>CODEX</th>
<th>Neutral</th>
<th>DAVINCI002</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abductive Reasoning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coherence with Premise</td>
<td>34%</td>
<td>48.5%</td>
<td>17.5%</td>
</tr>
<tr>
<td>Coherence with Ending</td>
<td>32%</td>
<td>42.5%</td>
<td>25.5%</td>
</tr>
<tr>
<td>Overall Coherence</td>
<td>40%</td>
<td>38%</td>
<td>22%</td>
</tr>
<tr>
<td>Counterfactual Reasoning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coherence</td>
<td>36.5%</td>
<td>39.5%</td>
<td>24%</td>
</tr>
<tr>
<td>Preservation</td>
<td>47.5%</td>
<td>39.5%</td>
<td>13%</td>
</tr>
</tbody>
</table>

leaving the target event function at the last.

Abductive Reasoning. Abductive reasoning requires models to generate a plausible hypothesis $H$ given the observations: premise $P$ and ending $E$. The chronological order of these three events is $P \rightarrow H \rightarrow E$, and the hypothesis causes the ending to occur.

In Figure 2, we regard the task definition as an instruction and place it as a comment at the beginning of the prompt. The causal structure is represented in the main function like: executing the premise, and if the hypothesis is met, executing the ending$^2$. The content of each event is presented as a comment of its function. The hypothesis function is placed at the last, leaving for models to complete. The generation process stops with a line break.

Counterfactual Reasoning. Counterfactual reasoning aims to rewrite a story under a counterfactual condition. As in Figure 1, the input consists of four parts: the premise $P$, the initial context $C_1$, the original ending $E_1$, and the counterfactual context $C_2$. Models are asked to generate the counterfactual ending $E_2$ that minimally modifies the original ending $E_1$ and is coherent with the counterfactual context $C_2$.

The causal relationships are represented with the if-elif structure. The premise $P$ is executed first, and then if the initial context $C_1$ is met, the original ending $E_1$ is executed; otherwise, if the counterfactual context $C_2$ is met, the counterfactual ending $E_2$ will be executed. For ease of exposition, we call the context hypothesis as well, being consistent with the former task. The event contents are also written as comments for event functions. We use $\#$ end to mark the finish of the ending.

3 Evaluation

Datasets. We experiment on the ART dataset (Bhagavatula et al., 2019) for the evaluation of abductive reasoning, and the TimeTravel dataset (Qin et al., 2019) for counterfactual reasoning, with 3,561 and 1,871 test instances, respectively.

Models. We experiment with CODEX (Chen et al., 2021), trained on a blend of code and text, as the Code-LLM. The specific version is code-davinci-002. We compare with two LLMs: the latest versions of GPT-3 (Brown et al., 2020) text-davinci-002 and text-davinci-003 (referred to as DAVINCI002 and DAVINCI003). Both of them originate from CODEX and are tuned with instructions. We follow OpenAI’s default settings in CODEX and DAVINCI decoding, and the text prompts for DAVINCI are in Figure A.1.

We also compare with previous unsupervised methods on these tasks, including DELOREAN (Qin et al., 2020), COLD (Qin et al., 2022), DIFFUSION (Li et al., 2022), CGMHI (Miao et al., 2019), and EDUCAT (Chen et al., 2022a)$^3$. Appendix A.3

$^2$Although not entirely accurate, this approximates the actual underlying causal relationships.

$^3$All these methods except DIFFUSION use GPT-2 (Radford et al., 2019) as the base model, and the model size ranges from medium to XL.
provides a brief introduction of these methods.

**Automatic Evaluation.** We use the following automatic evaluation metrics: BLEU$_4$ (Papineni et al., 2002), ROUGE$_L$ (Lin, 2004), CIDEr (Vedantam et al., 2015) and BERTScore (Zhang et al., 2019) based on BERT-base for abductive reasoning; BLEU$_4$, ROUGE$_L$ and BERTScore for counterfactual reasoning.

Table 1 reports the automatic evaluation results in the zero-shot setting. CODEX significantly outperforms previous methods and DA VINCI on both tasks (with significance level $\alpha = 0.01$), exhibiting strong causal reasoning ability. Although the two DA VINCI models are based on CODEX, their causal reasoning abilities may be weakened during instruction tuning, and this phenomenon is called alignment tax (Ouyang et al., 2022). DA VINCI003 underperforms DA VINCI002 on most metrics, probably because it tends to generate longer and more discursive outputs, which do not comply with the tasks.

**Human Evaluation.** We conduct pairwise comparison between CODEX and DA VINCI002 on 100 test examples. Annotators are asked to choose the better output given the task requirements. For abductive reasoning, the outputs are rated from three aspects: coherence with the premise, coherence with the ending, and the overall coherence. For counterfactual reasoning, the outputs are rated from coherence with the context and the extent of preserving the original ending. Each example is rated by at least two annotators, and the average interrater reliability is 0.64.

The results are shown in Table 2. CODEX outperforms DA VINCI002 in all aspects. It better considers the context in generation, and is able to preserve the original content in counterfactual reasoning.

**Contributions of the Model and the Prompt.** We exchange the prompts of code and text models, to measure the contributions of the model and the prompt. The results are in Table 3. We find that CODEX performs better with the code prompt, as the code prompt clearly describes the causal relation between events. Code prompts benefit the text model DA VINCI002 on abductive reasoning, but have negative impacts on counterfactual reasoning. A possible reason is that the causal structure in counterfactual reasoning is more complicated, leading to a more complex code which is harder for text models to understand.

### 4 What are Crucial in Code Prompts?

To paint a better picture of the key points in the code prompts, we intervene on the prompts from four aspects and measure the influences of the interventions. The four aspects we select are information, structure, format, and language. The former two, the prior information provided and the programming structure of functions, are content-related; the latter two, the code format and programming languages, are form-related. An ideal model should rely on the content and be insensitive to form perturbations. The interventions are described below, with examples in Figure A.2.

**Information.** We study two types of prior information: task instructions and function names. In No Instruction, we remove the task instruction from the prompts. In Function Name Perturbation, we replace original function names with anonymous functionX. For example, we replace premise() and hypothesis() in Figure 2 with functionA() and functionB(), respectively. It eliminates the information in function names and only allows models to learn the event relations from programming structures.

**Structure.** The first way to intervene in the programming structure is to convert the conditional structures into sequential structures, referred to as Sequential Structure. The events are executed sequentially, like premise(), hypothesis(),
ending() in abductive reasoning. In the second way called Disruption, we randomly disrupt the positions of the functions in the conditional structure. For instance, if hypothesis(): ending() can be disrupted into if ending(): hypothesis(). We also apply the function name perturbation in disruption to eliminate the impact of function names.

**Format.** We test three formats besides the original one: Class, Print and Return. The first one converts the original code into a class. We define the programming structure in the __init__ method, and move the event functions into the class. In Print, we represent the content of events as a string and print it in the function body, like def premise(): print("The Smiths ..."). And in Return, the string is the return value of event functions.

**Language.** We also convert the original Python programs into two other languages, Java and C, to evaluate the influence of programming languages.

**Intervention Results.** We evaluate the influence of interventions on abductive reasoning in Table 4, and the results on counterfactual reasoning are in Table A.2. The absence of prior information causes a small decrease in results. Even if the instruction or function names are not provided, CODEX is able to perform causal reasoning based on conditional statements. Changes in the programming structure have a larger negative impact. Comparing Function Name Perturbation and Disruption, the alteration of two characters (swap B and C in functionB and functionC) results in a major drop, showing that the conditional structure that reasonably depicts the relations between events is crucial in CODEX reasoning.

CODEX is quite robust towards format and language changes. Settings like Class and Java are even better than the original one, revealing that the performance can be further improved with delicate prompt engineering.

### 5 Conclusion

We investigate the causal reasoning ability of Code-LLMs. With code prompts of conditional statements, Code-LLMs achieve great performance in abductive and counterfactual reasoning, outperforming text-only LLMs significantly. Our study on different aspects of code prompts shows that providing a reasonable causal structure in code can help generate plausible outputs, and Code-LLMs are robust towards format perturbations.

### Limitations

**Language** Our experiments are conducted on English, as all Code-LLMs we know are pre-trained on English programming languages. Fundamentally, most popular programming languages are English-based, but international programming languages (which work in multiple languages) like Scratch, or non-English-based programming languages like Qalb also emerge. We look forward to the appearance of Code-LLMs on these programming languages.

**Prompt Engineering** We manually design the prompts without prompt engineering techniques such as prompt search. The searched prompts may outperform the ones we used, but our experiments on interventions show that CODEX is fairly robust towards format perturbations.

**Model** LLMs update quickly. From the time we submitted the paper until now, several new LLMs have been released. We try to compare their performance with ours. We select three new LLMs: CHATGPT, GPT-4 (OpenAI, 2023), and BARD4, and feed the text prompts to them. Because we do not have access to some of their APIs, we only experiment on a subset of 100 instances and report

---

**Table 4: Intervention results on abductive reasoning (%)**

<table>
<thead>
<tr>
<th></th>
<th>BLEU4</th>
<th>ROUGE4</th>
<th>CIDEr</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CODEX</strong></td>
<td>13.7</td>
<td>39.6</td>
<td>81.8</td>
<td>64.9</td>
</tr>
<tr>
<td><strong>Information</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Instruction</td>
<td>12.1</td>
<td>37.4</td>
<td>73.8</td>
<td>62.9</td>
</tr>
<tr>
<td>Function Name Perturbation</td>
<td>15.1</td>
<td>39.1</td>
<td>77.8</td>
<td>64.6</td>
</tr>
<tr>
<td><strong>Structure</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential Structure</td>
<td>9.6</td>
<td>36.8</td>
<td>72.0</td>
<td>63.5</td>
</tr>
<tr>
<td>Disruption</td>
<td>7.9</td>
<td>30.3</td>
<td>49.8</td>
<td>58.5</td>
</tr>
<tr>
<td><strong>Format</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>16.0</td>
<td>41.0</td>
<td>87.4</td>
<td>65.8</td>
</tr>
<tr>
<td>Print</td>
<td>13.8</td>
<td>39.4</td>
<td>82.0</td>
<td>65.0</td>
</tr>
<tr>
<td>Return</td>
<td>13.0</td>
<td>40.3</td>
<td>83.4</td>
<td>65.5</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>16.5</td>
<td>42.0</td>
<td>91.6</td>
<td>66.3</td>
</tr>
<tr>
<td>C</td>
<td>15.5</td>
<td>41.0</td>
<td>88.0</td>
<td>65.6</td>
</tr>
</tbody>
</table>

---

4Experiments are done with models updated to May 10, 2023.
<table>
<thead>
<tr>
<th></th>
<th>BLEU4</th>
<th>ROUGE_L</th>
<th>CIDEr</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CODEX</strong></td>
<td>15.0</td>
<td>39.8</td>
<td>82.2</td>
<td>67.8</td>
</tr>
<tr>
<td><strong>CHATGPT</strong></td>
<td>5.1</td>
<td>26.9</td>
<td>17.5</td>
<td>62.6</td>
</tr>
<tr>
<td><strong>GPT-4</strong></td>
<td>6.3</td>
<td>29.2</td>
<td>27.8</td>
<td>65.1</td>
</tr>
<tr>
<td><strong>BARD</strong></td>
<td>5.7</td>
<td>31.5</td>
<td>14.8</td>
<td>66.0</td>
</tr>
<tr>
<td><strong>CHATGPT</strong></td>
<td>15.3</td>
<td>34.7</td>
<td>70.0</td>
<td></td>
</tr>
<tr>
<td><strong>GPT-4</strong></td>
<td>38.5</td>
<td>55.5</td>
<td>78.6</td>
<td></td>
</tr>
<tr>
<td><strong>BARD</strong></td>
<td>12.1</td>
<td>22.0</td>
<td>62.1</td>
<td></td>
</tr>
</tbody>
</table>

(a) Abductive reasoning.

<table>
<thead>
<tr>
<th></th>
<th>BLEU4</th>
<th>ROUGE_L</th>
<th>CIDEr</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CODEX</strong></td>
<td>68.4</td>
<td>70.3</td>
<td>84.7</td>
<td></td>
</tr>
<tr>
<td><strong>CHATGPT</strong></td>
<td>15.3</td>
<td>34.7</td>
<td>70.0</td>
<td></td>
</tr>
<tr>
<td><strong>GPT-4</strong></td>
<td>38.5</td>
<td>55.5</td>
<td>78.6</td>
<td></td>
</tr>
<tr>
<td><strong>BARD</strong></td>
<td>12.1</td>
<td>22.0</td>
<td>62.1</td>
<td></td>
</tr>
</tbody>
</table>

(b) Counterfactual reasoning.

Table 5: Automatic evaluation results on a subset of 100 instances in the zero-shot setting. Numbers are in percentages (%).

the results in Table 5. CODEX outperforms all these models in the automatic evaluation, but part of the reason is that these models provide more detailed outputs than the reference. We provide a case study in Appendix A.5.

Since CODEX is no longer available to the public, we provide CODEX generation results in our GitHub repository. We also looked for alternatives and tried two open source Code-LLMs CODEGEN (Nijkamp et al., 2022) (version CodeGen-16B-Mono) and STARCODER (Li et al., 2023) with our code prompts. However, as shown in the case study, their performance is not comparable to CODEX, probably because they are more than ten times smaller in size.

**Ethics Statement**

Our work is based on off-the-shelf LLMs. As the results may inherit the underlying bias of LLMs, they cannot be used individually without human supervision. The Codex API was free when the experiments were conducted, and the Davinci APIs cost $0.02 per thousand tokens. We conduct all the experiments with less than $100. We recruit annotators for human evaluation from friends and colleagues of authors. All annotators are fairly paid with more than $10 per hour.

**Acknowledgments**

This work is supported in part by NSFC (62161160339). We would like to thank the anonymous reviewers for the helpful discussions and suggestions. For any correspondence, please contact Yansong Feng.

**References**


Bryan Rink, Cosmin Adrian Bejan, and Sandra Harabagiu. 2010. Learning textual graph patterns to detect causal event relations. In Twenty-Third International FLAIRS Conference.


A Appendix

A.1 Related Work

Causal Reasoning There is a growing interest in the NLP community to equip models with causal reasoning abilities. Chang and Choi (2005); Gordon et al. (2011) measure causality between words and phrases with statistical methods, Rink et al. (2010); Li and Mao (2019) use explicit semantic cues, and Liu et al. (2021); Zhang et al. (2022) discover causal relations with causal inference methods like propensity score matching. Li et al. (2019) finetune LLMs on causal event corpus, and Du et al. (2021); Wang et al. (2022a) augment LLMs with causal knowledge graphs. Contrast to them, we explore the causal reasoning abilities acquired by Code-LLMs in pre-training.

Applying Code-LLMs to Natural Language Tasks With the recent development of Code-LLMs, several works attempt to solve natural language tasks with code models. They mainly focus on two areas: numerical reasoning and structural prediction. Gao et al. (2022); Chen et al. (2022b); Wu et al. (2022) apply Code-LLMs to numerical reasoning. They generate programs with Code-LLMs and feed the programs into an external interpreter to derive the answer. Madaan et al. (2022); Wang et al. (2022b) leverage the text-to-structure translation ability of Code-LLMs to perform structural prediction tasks. They ask models to generate structures in the form of code, and convert the generated code into the task output format. In addition, Hu et al. (2022) takes advantages of Code-LLMs on text-to-SQL generation. Different from them, we leverage the causal reasoning ability of Code-LLMs, and ask them to generate natural language events given the causal structure.

A.2 Prompts

Figure A.1 demonstrates the prompts of probing DAVINCI. Specifically, the language conversion is made automatically by CODEX with the instruction # python to java/c. Figure A.2 shows the interventions on code prompts for abductive reasoning.

A.3 Models for Comparison

We compare with previous unsupervised methods on the two tasks, including DELOREAN (Qin et al., 2020), COLD (Qin et al., 2022), and DIFFUSION (Li et al., 2022) on abductive reasoning; and CGMH (Miao et al., 2019), EDUCAT (Chen et al., 2022a), DELOREAN, and COLD on counterfactual reasoning. Among them, DELOREAN and COLD are constraint-based models. They regard the task requirements as constraints (for example, the generated text should be consistent with the premise, and coherent with the ending in the abductive reasoning task), and iteratively update text representation to meet the constraints. CGMH and EDUCAT are editing-based models targeted for counterfactual reasoning. They start from the original ending and edit it to meet the counterfactual context. DIFFUSION builds a controllable LM based on continuous diffusions to perform control tasks including abductive reasoning.

A.4 Additional Results

<table>
<thead>
<tr>
<th></th>
<th>Min-Edit</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELOREAN</td>
<td>52.9</td>
<td>73.7</td>
</tr>
<tr>
<td>COLD</td>
<td>56.8</td>
<td>73.5</td>
</tr>
<tr>
<td>CODEX</td>
<td>58.0</td>
<td>79.5</td>
</tr>
</tbody>
</table>

Table A.1: Counterfactual reasoning results in the first-sentence setting (%).

First-Sentence Setting of Counterfactual Reasoning Endings in the original counterfactual reasoning data TimeTravel are of three sentences. Due to the computation constraint of COLD (Qin et al., 2022), it is evaluated in a first-sentence setting: only the first sentence of the original ending is used, and models are asked to generate a one-sentence counterfactual ending. We conduct experiments in the first-sentence setting with the metrics used in Qin et al. (2022). As shown in Table A.1, CODEX outperforms previous methods in this setting.

Intervention on Counterfactual Reasoning Table A.2 demonstrates the intervention results on counterfactual reasoning. The observations are similar to those in the abductive reasoning task: changes in the programming structure affect CODEX’s performance largely, changes in the information affect less, and CODEX is robust towards format and language changes.

One-shot Setting We also conduct experiments in the one-shot setting. Models are shown with one demonstration example in the in-context learning manner, and the example is identical among the models. As shown in Table A.3, both DAVINCI002 and CODEX are better than in the...
**Abductive Reasoning**

Generate a plausible explanatory hypothesis given the premise and the ending.

**Premise:** The Smiths were having holidays done of the children.

**Ending:** Ty’s face lit up as he ran to the new toy, happily posing for photos.

**Hypothesis:** The Smiths were having holidays.

---

**Counterfactual Reasoning**

Given an original story and an intervening counterfactual event, the task is to minimally revise the story to make it compatible with the given counterfactual event.

**Premise:** Janice was excited to bring cupcakes to her work for her birthday.

**Initial event:** She worked all day on making the perfect frosting.

**Original ending:** Each cupcake was truly a work of art. Everyone at her work loved them. Janice was thrilled and happy for the rest of the day.

**Counterfactual event:** She completely rushed making the frosting.

**New ending:** Each cupcake was a mess. The frosting was lumpy and tasted terrible. Janice was embarrassed and felt terrible for the rest of the day.

---

### Table A.2: Intervention results on counterfactual reasoning (%)

<table>
<thead>
<tr>
<th></th>
<th>BLEU₄</th>
<th>ROUGE₄</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CODEX</strong></td>
<td>66.8</td>
<td>70.0</td>
<td>82.5</td>
</tr>
<tr>
<td><strong>Information</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Instruction</td>
<td>55.4</td>
<td>60.1</td>
<td>77.0</td>
</tr>
<tr>
<td>Function Name Perturbation</td>
<td>65.4</td>
<td>69.0</td>
<td>82.2</td>
</tr>
<tr>
<td><strong>Structure</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential Structure</td>
<td>43.4</td>
<td>50.2</td>
<td>68.2</td>
</tr>
<tr>
<td>Disruption</td>
<td>16.0</td>
<td>23.5</td>
<td>55.2</td>
</tr>
<tr>
<td><strong>Format</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>63.6</td>
<td>67.4</td>
<td>81.1</td>
</tr>
<tr>
<td>Print</td>
<td>73.3</td>
<td>74.7</td>
<td>85.3</td>
</tr>
<tr>
<td>Return</td>
<td>69.4</td>
<td>70.5</td>
<td>83.0</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>71.1</td>
<td>73.5</td>
<td>84.5</td>
</tr>
<tr>
<td>C</td>
<td>71.9</td>
<td>74.2</td>
<td>85.0</td>
</tr>
</tbody>
</table>

---

**A.5 Case Study**

We randomly select some generation examples and demonstrate them in Table A.4. Comparing CODEX and DAVINCI, CODEX generations are more coherent with the context, while DAVINCI sometimes cannot take into account the premise. CODEX also understands the task instruction well and better preserves the original ending in counterfactual reasoning. Generations of more powerful LLMs like CHATGPT and GPT-4 are coherent with the context, but they add much detail and barely keep the original ending. Although open source Code-LLMs like CODEGEN and STAR-CODER can follow the code prompts and generate sentences in the required format, most of their outputs are inconsistent with the premise and the ending.
Table A.3: Evaluation results in the one-shot setting (%).

<table>
<thead>
<tr>
<th></th>
<th>BLEU_4</th>
<th>ROUGE_L</th>
<th>CIDEr</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAVINCI002</td>
<td>8.2</td>
<td>33.5</td>
<td>55.9</td>
<td>61.7</td>
</tr>
<tr>
<td>CODEX</td>
<td>17.9</td>
<td>42.3</td>
<td>91.7</td>
<td>67.1</td>
</tr>
</tbody>
</table>

(a) Abductive reasoning.

Table A.4: Examples of model generations.
Figure A.2: Examples of code prompt interventions in abductive reasoning.

9020
A For every submission:

☑ A1. Did you describe the limitations of your work?
   Limitation Section

☑ A2. Did you discuss any potential risks of your work?
   Ethics Statement

☑ A3. Do the abstract and introduction summarize the paper’s main claims?
   1. Introduction

☒ A4. Have you used AI writing assistants when working on this paper?
   Left blank.

B ☑ Did you use or create scientific artifacts?

   3. Evaluation

☑ B1. Did you cite the creators of artifacts you used?
   3. Evaluation

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
   Not applicable. In the supplementary data

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
   3. Evaluation

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
   Not applicable. The data we use is created and checked by previous work.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
   Limitation Section

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
   3. Evaluation

C ☑ Did you run computational experiments?

   3 & 4

☐ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
   Not applicable. The parameters and computational budget are not public available.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?

Limitation Section

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?

Evaluation

C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

In the supplementary code

D  ✓ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Appendix A.4

D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?

The instructions are briefly introduced in Appendix A.4

D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?

Ethics Statement

D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?

Appendix A.4

D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?

Not applicable. Ethics review is not required.

D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?

Not applicable. Ethics Statement