

©Copyright 2024

Chenxi Li

A Versatility Analysis: Investigating Large Language Models' Performance Beyond Conventional Benchmarks

Chenxi Li

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2024

Reading Committee:

Fei Xia

Shane Steinert-Threlkeld

Program Authorized to Offer Degree:
Computational Linguistics

University of Washington

Abstract

A Versatility Analysis: Investigating Large Language Models' Performance Beyond Conventional Benchmarks

Chenxi Li

Chair of the Supervisory Committee:

Fei Xia

Department of Linguistics

Recent progress in large language models (LLMs) has marked a notable milestone in the field of artificial intelligence. The conventional evaluation of LLMs primarily relies on existing tasks and benchmarks, raising concerns about test set contamination and the genuine comprehension abilities of LLMs. This study introduces a novel approach by developing unique datasets to circumvent potential data contamination issues and scrutinizes LLMs' adaptability to new tasks, their sensitivity to prompt variations, and their error tendencies. We investigate the capacity of LLMs to adapt to new but simple tasks, especially when they diverge from the models' pre-existing knowledge. Our methodology emphasizes the creation of straightforward tasks, facilitating a precise error analysis to uncover the underlying causes of LLM failures. This strategic approach also aims to uncover effective strategies for enhancing LLM performance based on the detailed error analysis of system output.

TABLE OF CONTENTS

	Page
List of Figures	iv
List of Tables	v
Chapter 1: Introduction	1
Chapter 2: Related Work	3
2.1 Evaluations on existing benchmarks	3
2.2 Development of new benchmarks	4
2.3 Evaluation on Robustness	4
2.4 Evaluation challenges	5
Chapter 3: Methodology	7
3.1 Task design	7
3.2 Dataset creation	7
3.3 Prompt design	7
3.4 Evaluation	8
3.5 Error analysis	8
3.6 Post error analysis CoT prompt design	8
Chapter 4: Task Design	10
4.1 Task 1 & 2: Arithmetic Expression Evaluation with Changed Precedence	10
4.2 Task 3 & 4: Machine Translation with Artificial Language	11
4.3 Task 5 & 6: Basic Deduction with a Twist	13
Chapter 5: Dataset creation	16
5.1 Task 1	16
5.2 Task 2	17
5.3 Task 3	17
5.4 Task 4	18

5.5	Task 5	18
5.6	Task 6	19
Chapter 6:	Prompt design	20
6.1	Task 1	20
6.2	Task 2	22
6.3	Task 3	28
6.4	Task 4	34
6.5	Task 5	35
6.6	Task 6	36
Chapter 7:	Experiments	38
7.1	Task 1: Arithmetic Expression Calculation	38
7.2	Task 2: Choosing Precedence of Operators	39
7.3	Task 3: Machine Translation	39
7.4	Task 4: Inferring Word Order	44
7.5	Task 5: Basic Deduction	44
7.6	Task 6: Deduction with Different Types of Predicates	45
7.7	Comparison of GPT4 and LLaMA	46
Chapter 8:	Error Analysis	48
8.1	Q1: How effectively can LLMs adapt to new tasks?	48
8.2	Q2: How robust are LLMs to minor variations of prompts?	49
8.3	Q3: What are the common error types made by LLMs in our designed tasks?	51
8.4	Q4: What strategies can be employed to enhance LLM performance on our tasks?	52
Chapter 9:	Discussion	54
9.1	Limitations	54
9.2	Suggestions on evaluating LLMs	55
Chapter 10:	Conclusion	56
Appendix A:	GPT4 outputs on four Types of Prompts in Task 1	63
A.1	0-shot output	63
A.2	0-shot-CoT output	63
A.3	1-shot output	64
A.4	1-shot-CoT output	64

Appendix B: GPT4 outputs on four Types of Prompts in Task 2	66
B.1 0-shot output	66
B.2 1-shot-CoT-simple output	68
B.3 1-shot-CoT-partial output	69
B.4 1-shot-CoT-full output	71
Appendix C: GPT4 outputs on four Types of Prompts in Task 3	73
C.1 0-shot output	73
C.2 0-shot-CoT output	74
C.3 1-shot output	75
C.4 1-shot-CoT output	76

LIST OF FIGURES

Figure Number	Page
4.1 Redefine priorities of operators in Task 1-2 by moving operators.	11
8.1 Raw count with regard to error types across No CoT, CoT1, and CoT2 conditions for Task 1. The explanation of five types is in Section 8.3	53

LIST OF TABLES

Table Number	Page
4.1 Example prompt for Task 1-2.	11
4.2 The set of rules that a prompt can select when defining Language-A for Task 3. . .	12
4.3 Example prompt for Task 3 and Task 4.	13
4.4 Four types of predicates and examples of definition given in Task 6 prompts. . . .	14
4.5 Example prompt for Task 5 and 6. Task 5 uses the bAbI dataset (Weston et al., 2015); Task 6 uses our own newly created datasets.	15
7.1 The accuracy of GPT4-Turbo on Task 1. “AP” denotes the approach, where “s” and “c” denote “ <i>shot</i> ” and “ <i>CoT</i> ”, respectively (e.g., “1-s-c” means 1-shot-CoT). Col 1D-5D refers to the numbers of digits in operands, ranging from 1 to 5. Arithmetic expressions used in each column within (a) and (b) are the same, but the priorities of the operators in the expressions are different between no move, one move, and two moves. The system performance decreased with the increase of digits. In some cases, no move expressions have a lower accuracy than one-move or two-move expressions of 1-s-c prompt. This might be caused by the fact that when digit is larger than 3, the system makes many calculation mistakes, but 1-s-c is not designed at correct calculation mistakes.	40
7.2 Performance of GPT4 on Task 2 under two-move configuration. Each cell has the format x / y , where x is the system accuracy and y is the percentage of <i>invalid</i> answers (i.e., GPT4 does not pick any choice).	41
7.3 The performance of GPT4 on Task 3 with 0-shot prompts. ‘S’ and ‘M’ represent the number of syntactic and morphological rules included in the prompts. ‘EM’ is exact match, ‘F1’ is Morph F1, and for METEOR, we report the METEOR score, the number of aligned chunks C and the number of unaligned target words UA . . .	42
7.4 The performance of GPT4 on Task 3 with four types of prompts	43
7.5 GPT4’s accuracy on Task 4. See Section 3.4 for the meaning of the row and column headers.	44
7.6 Accuracy of GPT4 on Task 5. ‘Base’: original dataset from bAbI; ‘Ran’: replace nouns with random strings; ‘Neg’: add ‘not’ to the predicate; ‘Quan’: add ‘some’ to the subject of the sentence; ‘ext’: extend the reasoning chain.	45

7.7	Accuracy of GPT4 on Task 6. ‘T’: transitive predicate; ‘S’: symmetrical predicate. ‘NT’: none-transitive predicate. ‘NS’: none-symmetrical predicate. ‘random predicate’ is when the predicate is replaced by random strings in the statements; ‘random Entity’ is when the entities are replaced by random strings in the statements.	45
7.8	Accuracy comparison between GPT4 and LLaMA on Task 1 for two-digit expressions	46
7.9	Accuracy comparison between GPT4 and LLaMA on Task 5; column title adopted from Table 7.6	46
7.10	Accuracy comparison between GPT4 and LLaMA on Task 6; column title adopted from Table 7.7	47
7.11	Comparative analysis of GPT-4 and LLaMA performance on Task 4 for original and switch experiments	47
8.1	Raw count of correct and incorrect predictions made by GPT4 on the original dataset for Task 5 when one evidence statement is factual vs. counterfactual.	48
8.2	Confusion Matrix for Task 4’s 2-Order Experiment Focused on SOV Rule. Rows represent the correct answers, while columns indicate GPT4’s predictions.	49
8.3	Confusion Matrix for Task 4’s 2-Order Switch Experiment asking about canonical word order. Rows represent the correct answers, while columns indicate GPT4’s predictions.	50
8.4	Frequency table of SOV Rule Recognition Before and After Sentence Pair Reordering in Tables 8.2 and 8.3. ‘SOV 1st’ indicates scenarios where the first sentence pair follows the SOV structure, and conversely for the switched condition. P-value is less than 0.00001.	50
8.5	The count of correct and incorrect instances out of 250 with regard to the position of the evidence in the prompt	51
8.6	Error Type Distribution for ‘PP Before Noun’ Rule in Task 4: Analyzing 200 Instances. ‘MR’ denotes misremembered translations, with MR1 and MR2 indicating errors in the first and second sentence pairs, respectively. ‘FO’ signifies incorrect observations, distinguishing between the first (FO1) and second (FO2) translated sentences.	52

ACKNOWLEDGMENTS

This thesis is an extension of the paper ‘Challenging Large Language Models with New Tasks: A Study on their Adaptability and Robustness’ submitted to ACL 2024. The submission is a joint work with Yuanhe Tian, Zhaxi Zerong, Yan Song, and Fei Xia. I am involved in the design of all six tasks. I have implemented the automatic dataset generation and task evaluation for Task 1, Task 2, Task 5, and Task 6; I have created the requisite corpora for dataset generation in Task 3 and Task 4. I extend gratitude to Yuanhe, who ran the experiments on local models and contributed greatly to the design of Task 1, Task 2, and Task 3; to Zhaxi, who implemented automatic dataset generation, evaluation, and error analysis for Task 3 and Task 4. and to Yan, who provided the financial supports for all the experiments.

I thank Shane for agreeing to be on the reading committee and for providing valuable feedback on this thesis.

Lastly, I extend my heartfelt gratitude to Fei Xia, whose support has been instrumental throughout the entire duration of the joint work and this thesis. She hosted numerous meetings for the joint work where we collaborated on brainstorming designs and meticulously analyzed errors across all tasks. Her guidance for this thesis was invaluable, ranging from shaping the big-picture framework to refining the smallest nuances in wording. Thank you, Fei, for your help and support at a moment’s notice.

Chapter 1

INTRODUCTION

LLMs have produced impressive results on many NLP tasks and existing studies for evaluating LLMs mainly leverage benchmark datasets for existing tasks, such as question answering (Berant et al., 2013; Joshi et al., 2017; Kwiatkowski et al., 2019; Reddy et al., 2019), reasoning (Clark et al., 2018), and math word problems (Hendrycks et al., 2021a; Cobbe et al., 2021b; Shi et al., 2022). These evaluation approaches face the test set contamination issue (Oren et al., 2023). Data contamination is when test data are seen by LLMs during pre-training, inflating the model performance. Therefore, it is not clear if the high performance of LLMs results from the ability to correctly answer questions in the existing benchmark or relying on the memorization of prior knowledge during training.

Another challenge with prior research lies in the increasing complexity of benchmark tasks. Paired with the frequent absence of thorough error analysis, the resulting performance metrics often amount to mere numbers. They lack the substantive insights necessary for understanding why models struggle with certain tasks, which is essential for improving LLMs in the future. In this study, we create new tasks and new datasets instead of exploiting current benchmarks to reduce the risk of data contamination, addressing the following research questions:

- (Q1) How effectively can LLMs adapt to new tasks, particularly when instructions in the prompts conflict with the model’s prior knowledge?
- (Q2) How robust are LLMs to minor variations of prompts?
- (Q3) What are the common error types made by LLMs in our designed tasks?
- (Q4) Drawing from the insights gained in the preceding questions, what strategies can be employed to enhance LLMs’ performance on specific tasks?

Importantly, our tasks are designed to be novel but simple to facilitate identifying the source of LLMs’

deficiencies during error analysis. In addition, based on detailed error analysis, we improve system performance via techniques such as chain of thought (Wei et al., 2022) prompts.

Chapter 2

RELATED WORK

LLMs have exhibited unparalleled performance across various tasks, garnering significant attention in both academic and industrial circles. This surge in popularity has prompted a growing emphasis on the evaluation of LLMs in recent years. Numerous studies of different approaches have been undertaken to assess LLMs. Some aim to comprehensively evaluate LLMs by scaling across multiple popular benchmarks, while others concentrate on scrutinizing specific abilities of LLMs through the introduction of newly developed datasets, underscoring the importance of such data. Additionally, there are studies that delve into exploring the versatility and robustness of LLMs, particularly in response to minor variations in prompts

2.1 Evaluations on existing benchmarks

In previous years, researchers have developed datasets that have earned widespread recognition within the field. For instance, benchmarks like GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) have become pivotal in evaluating models' proficiency in natural language understanding, while GSM8K (Cobbe et al., 2021a) and MATH (Hendrycks et al., 2021b) are frequently cited when assessing the mathematical capabilities of LLMs. Despite being established prior to the emergence of numerous new models, these benchmarks continue to serve as foundational pillars for evaluating model performance.

As a result, some papers now concentrate on assessing the efficacy of new models using these established benchmarks (Laskar et al., 2023; Bang et al., 2023; Qin et al., 2023). By evaluating these new models against classic benchmarks, they aim to provide valuable insights into the comparative strengths and weaknesses of different models. At the same time, these works also serve as a record for future reference and thus encourage competition and innovation.

2.2 Development of new benchmarks

Previous benchmarks do provide a solid ground for the evaluation of LLMs, but some researchers have identified gaps in current datasets and proposed new benchmarks to address these limitations. For example, GHOSTS and miniGHOSTS (Frieder et al., 2023) were introduced to assess graduate-level mathematical abilities holistically, as previous benchmarks in this domain were deemed either too simplistic or too small in scale.

In the realm of reasoning, LogiEval (Liu et al., 2023) was created by combining newly created out-of-distribution datasets with existing logic reasoning benchmarks such as LogiQA (Liu et al., 2020), aiming to evaluate LLMs’ reasoning robustness to novel data. Likewise, NeuRL (Xu et al., 2023) was designed to gauge pure reasoning ability through content-neutral reasoning questions, driven by concerns that model performance could be influenced by knowledge bias.

These new benchmarks proposed to evaluate LLMs allows researchers to address limitations in existing datasets and provide comprehensive assessments of LLM capabilities in specific domains or tasks.

2.3 Evaluation on Robustness

While a significant body of research is devoted to evaluating LLMs on benchmarks, both existing and new, some studies diverge from benchmark-centric approaches to explore the versatility of LLMs in handling different variations of questions. Considerable attention has been focused on evaluating their resistance to adversarial attacks (Zhao et al., 2021; Hou et al., 2024). Also, LLMs might be more robust than anticipated to prompts containing demonstration examples of poor quality. For example, CoT with invalid demonstrations (Wang et al., 2023) and in-text learning demonstrations with random labels (Min et al., 2022) can both notably improve model performance.

A related study explores the effect of reordering answer choices in multiple-choice questions on model performance (Pezeshkpour and Hruschka, 2023). However, our approach differs in that we maintain the answer choices for tasks in the format of multiple choice question, but alter the placement of key information within the question to test if LLMs maintain a similar performance.

2.4 Evaluation challenges

While Large Language Models (LLMs) have demonstrated remarkable capabilities that surpass even fine-tuned State-of-the-Art (SOTA) models in specific domains such as mathematics (Bubeck et al., 2023), reasoning, as suggested by some studies (Gendron et al., 2024; Xu et al., 2023), seems to be an aspect where LLMs do not consistently outperform smaller models, especially when they are assessed on out-of-distribution (OOD) datasets (Liu et al., 2023). Notably, many previous studies have not incorporated human evaluations for comparison. Consequently, the conclusions drawn regarding the competence of LLMs in certain tasks may lack real-world significance. Without a direct performance comparison between LLMs and humans, achieving high scores in benchmarks does not necessarily justify the practical implementation of LLMs in those specific areas.

At the same time, even in experiments where LLMs exhibit remarkable performance, the results may be influenced by the data contamination problem (Brown et al., 2020). Given the staggering size of pre-training data, the likelihood of test instances bearing a resemblance discernible to LLMs is high. This is especially true for datasets that have been available online for an extended period.

The inspection of the source and content of the Colossal Clean Crawled Corpus (Raffel et al., 2023), upon which some LLMs are trained, reveals that it encompasses evaluation instances for certain NLP tasks (Dodge et al., 2021). Data contamination could potentially lead to inflated evaluation results as LLMs are shown to be capable of generating correct answers through memorization (Magar and Schwartz, 2022).

In addressing this concern, various approaches have been proposed and developed to identify data contamination, focusing on the degree of exposure of test instances during training (Oren et al., 2023; Golchin and Surdeanu, 2023). Caution is warranted when reporting evaluation results. In some studies, the results that could potentially be skewed due to data contamination are flagged so that readers can use them with caution (Bubeck et al., 2023).

Several studies have integrated strategies to mitigate such risks in dataset development. For instance, some devised counterfactual tasks that are less likely to have been encountered by LLMs (Wu et al., 2023), while others constructed datasets featuring false and fictional ontologies, which are potentially unfamiliar to LLMs (Saparov and He, 2023). While our work draws some inspiration on such studies, our work differs from theirs in two ways. Firstly, our novel tasks aim to evaluate different

domains than theirs with simple and straightforward tasks. Secondly, we conduct a comprehensive error analysis to identify common error types made by LLMs on these novel tasks.

Chapter 3

METHODOLOGY

To answer our research questions, we followed the five steps to create six tasks: task design, data creation, prompt design, evaluation, and error analysis. For the first three tasks, we also further improve the prompt by adding relevant CoT aimed at solving problems we have discovered in error analysis. The improved CoT prompts have greatly improved the results.

3.1 Task design

The core of our research idea is to design tasks that challenge LLMs. Our tasks are designed to be challenging not because they require advanced knowledge or skills. Instead, they are intentionally designed to be simple. The reason is that complex tasks make it hard to pinpoint where LLMs struggle during error analysis. The tasks are challenging in that they introduce novel information in the instructions, something LLMs might not have encountered during pre-training. Also, to ensure practical relevance, we design the tasks based on real-life problems, aligning them with potential real-world applications. The design details of each task is in Chapter 4.

3.2 Dataset creation

After task design, we manually wrote an original prompt for each task. We then use these original prompts as templates and write Python script to generate the whole datasets for each task. Both the datasets and the gold standards are generated automatically. The details on the dataset creation is in Chapter 5.

3.3 Prompt design

Previous studies showed that even small changes in the prompt can significantly affect results(Radford et al., 2021). Therefore, the next step is that we fine-tune the original prompts to ensure LLMs produce optimal results through small-scale experiments during which we select ten to twenty

instances, evaluate them, and examine the system outputs. We experiment with different wordings for each prompt, selecting the one that consistently produced reliable outputs. For instance, in Task 1, we initially restricted LLMs to providing only numerical answers for arithmetic expressions. However, this led to poor performance. Removing this restriction allowed LLMs to generate task-solving steps freely, resulting in improved accuracy. Another crucial aspect of conducting meticulous, small-scale examinations of system output lies in its facilitation of evaluation script development. LLMs can generate responses that surpass prediction and expectation. By guiding LLMs towards generating content in desired formats, it becomes feasible to implement evaluation procedures effectively. The examples of finalized prompt for each task are presented in Chapter 6.

3.4 Evaluation

After finalizing the prompts for each task, we proceed to develop Python scripts and conduct experiments to generate system outputs from three distinct LLMs. Subsequently, we develop Python scripts to evaluate these system outputs against the gold standard. The experimental setups for each task and the detailed results for three LLMs are elaborated in Chapter 7.

3.5 Error analysis

Due to time and resource constraints, we could not conduct an exhaustive error analysis on the outputs of every dataset. However, we still performed a significant amount of analysis on Task 1, Task 3, and Task 5, and some on the other three tasks. The selection of error analysis is guided by two main criteria. Firstly, if one dataset showed notably lower performance compared to others it was being compared with, we conducted an error analysis to understand the underlying reasons behind the major drop. Secondly, if a dataset that was conceptually easier yielded unexpectedly low or similar performance to others, we also investigated this abnormality through error analysis. The findings of the error analysis are presented in Chapter 8

3.6 Post error analysis CoT prompt design

The extensive error analysis across tasks have allowed us to discover some common types of error made by LLMs. We then design CoT instructions aimed at eliminating these error types and insert

them into the prompts to further test if such instructions could help models to rectify the mistake and achieve the correct answer. The details and justification on the CoT instructions are discussed in Chapter 6; the effectiveness of these CoT instructions is discussed in Chapter 8.

Chapter 4

TASK DESIGN

To answer our research questions, we have formulated six distinct tasks. By crafting corresponding datasets and prompts, we ensured comprehensive coverage and contextually relevant responses from the language model. Furthermore, we also perform detailed error analyses across tasks to provide insights into its performance and limitations.

4.1 Task 1 & 2: Arithmetic Expression Evaluation with Changed Precedence

Previous studies (Brown et al., 2020; Anil et al., 2023) have tested LLMs’ performance in arithmetic expression evaluation, revealing that chain-of-thought (CoT) (Wei et al., 2022) enhances model accuracy. To answer Q1, we change the original task by redefining the precedence of elementary arithmetic operations to assess LLMs’ ability with changed math rules. In Task 1, the LLM is given the new precedence and an expression and asked to produce the correct value; In Task 2, the model is given an expression and its corrected value and asked to choose the correct precedence of operators. The original prompts for the tasks are shown in Table 4.1. Subsequently, we analyze the system output to investigate the effect of variations in prompts on model performance, addressing Q2-Q4.

We measure the difference between the new and the standard precedence of expressions by the number of ‘moves’ needed to change from the latter to the former precedence. The precedence is represented in a five-slot array. Fig 4.1(a) shows the standard precedence, where ‘*’ and ‘/’ are in slot 2, and ‘+’ and ‘-’ are in slot 4. To create a new precedence, we move some operators from their original slots to new ones. For instance, in Fig 4.1(b), ‘+’ is moved from slot 4 to 1; in Fig 4.1(c), ‘+’ is from slot 4 to 1, and ‘*’ from slot 2 to 3. We call these three cases **no move**, **one move**, and **two moves**, respectively. In Section 3.1, we will report system performance with respect to the number of moves.

(a) No Move: ‘ * ’= ‘ / ’ > ‘ + ’ = ‘ - ’	1	2	3	4	5
		* /		+ -	

(b) One Move: ‘ + ’ > ‘ * ’ = ‘ / ’ > ‘ - ’	1	2	3	4	5
	+	* /		-	

(c) Two Move: ‘ + ’ > ‘ / ’ > ‘ * ’ > ‘ - ’	1	2	3	4	5
	+	/	*	-	

Figure 4.1: Redefine priorities of operators in Task 1-2 by moving operators.

Task 1

We redefine the priorities of elementary arithmetic operators as follows: ‘+’ > ‘*’ = ‘-’

Based on the new priorities, what is the value of 8 - 4 * 4 + 9? You can give the result as a decimal, and please put the final numeric result in brackets [], such as [1.0]

Task 2

For the expression 8 - 4 * 4 + 9. the precedence of the operators is redefined. If 8 - 4 * 4 + 9 = 52, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].

(A) ‘+’ = ‘*’ = ‘-’ (B) ‘-’ = ‘*’ > ‘-’ (C) ‘+’ > ‘*’ = ‘-’ (D) ‘-’ = ‘*’ = ‘+’.

Table 4.1: Example prompt for Task 1-2.

4.2 Task 3 & 4: Machine Translation with Artificial Language

Prior research on machine translation (MT) primarily focus on translation between natural languages such as English and French (Bawden and Yvon, 2023; Tamura et al., 2023; Zhang et al., 2023). We choose to use a simple artificial language, Language-A, to probe into LLMs’ understanding of prompts so that LLMs cannot rely on the translation pairs of existing languages that they have seen during pre-training. In Task 3, LLMs are given an English sentence, the translation of the words, syntactic and morphological rules of Language-A selected from Table 4.2 and asked to translate the English sentence into Language-A. Task 4’s prompts include two English sentences and their

ID	Rule
S1 (SOV)	In Language-A, the canonical word order of sentences is Subject-Object-Verb.
S2 (PV)	In Language-A, modifiers of verbs come before the verbs. If a verb is modified by both a preposition phrase (PP) and an adverb, the correct word order is PP first, then the adverb, and then the verb.
S3 (PN)	In Language-A, modifiers of nouns appear before nouns. When a noun has multiple modifiers, the word order is prepositional phrases first, then determiners, then adjectives, and finally the noun.
M1 (Plu)	In Language-A, a singular noun and a plural noun start with prefix ‘S_’ and ‘P_’, respectively, followed by its lemma.
M2 (PT)	In Language-A, verbs in past tense and present tense are represented by the lemma, followed by suffix ‘_-’ and ‘_+’, respectively.
M3 (AR)	In Language-A, each adjective lemma is repeated twice consecutively in a sentence.

Table 4.2: The set of rules that a prompt can select when defining Language-A for Task 3.

translations in Language-A as well as the word-level translation pairs. LLMs are asked to infer Language-A’s syntactic rules (e.g., whether Language-A is SOV or SVO). The original prompts are shown in Table 4.3.

The tasks assess LLMs’ prior knowledge about linguistics (e.g., the meaning of SVO) and their adaptability to new linguistic challenges such as translating English into an artificial language, addressing research question Q1. We explore LLMs’ robustness with respect to prompt variations (Q2), identify common error types (Q3), and improve LLMs’ performance with CoT based on the error analysis (Q4).

Task 3

The following are the linguistic rules of Language-A: <Rules>

For anything not explicitly specified, Language-A follows the same word order as English.

Here are the word pairs between English and the Language-A: <Word Pairs>.

Translate the following simple sentence into Language-A: ‘<English Sentence>’.

Please put the final translated result in brackets [] at the end of your response.

Task 4

<Sentence X1> in English is translated to <Sentence Y1> in Language-A <Sentence X2> in English is translated to <Sentence Y2> in Language-A

Here are the word pairs between English and Language-A: <Word Pairs>.

What is the canonical word order in Language-A: (a) SVO only, (b) VSO only, (c) SVO or SOV, (d) SOV only

Table 4.3: Example prompt for Task 3 and Task 4.

4.3 Task 5 & 6: Basic Deduction with a Twist

In Task 5, we evaluate LLMs’ reasoning ability on the bAbI dataset (Weston et al., 2015), which has been evaluated by several studies (Xu et al., 2023; Bang et al., 2023). We adopt the basic deduction test set from bAbI; an example is shown in Table 4.5. The example prompt contains eight statements: (1) Wolves are afraid of mice. (2) Sheep are afraid of mice. (3) Winona is a sheep. (4) Mice are afraid of cats. (5) Cats are afraid of wolves. (6) Jessica is a mouse. (7) Emily is a cat. (8) Gertrude is a wolf. The question is ‘What is Emily afraid of?’ The correct answer give by the bAbI dataset is ‘wolf/wolves’ using statements (5) and (7). To answer Q1, we modify the dataset in two ways: we replace nouns in the original statements with random strings to test LLMs’ ability to handle unfamiliar entities; to answer Q2; we negate the verb or add *some* to the subject or object of some statements to check whether LLMs interpret negation and quantifiers properly during reasoning.

In Task 6, we assess LLMs’ capability to discern semantic nuances of predicates. Some predicates represent transitive relations (e.g., if X runs faster than Y and Y runs faster than Z, then X runs faster than Z); Some represent symmetric relations (e.g., ‘X equals Y’ implies ‘Y equals X’). Thus,

Predicate Types	Example of Definition
None-transitive & None-symmetrical	X fears Y
Transitive & None-symmetrical	X runs faster than Y
None-transitive & Symmetrical	X and Y are partially related
Transitive & Symmetrical	X and Y have the same speed

Table 4.4: Four types of predicates and examples of definition given in Task 6 prompts.

predicates can be divided into four types according to these two properties, as shown in Table 4.4. We want to test whether LLMs can distinguish different types of predicates during reasoning. To prevent LLMs from utilizing their prior knowledge about the actual predicates in English, we define new predicates (written as random strings) in the prompts and ask LLMs to reason about the predicates. The original prompts of Task 5-6 are in Table 4.5.

Task 5

(1) Wolves are afraid of mice. (2) Sheep are afraid of mice. (3) Winona is a sheep. (4) Mice are afraid of cats. (5) Cats are afraid of wolves. (6) Jessica is a mouse. (7) Emily is a cat. (8) Gertrude is a wolf.

Based on the above sentences, what is Emily afraid of?

(a) cat (b) mouse (c) wolf (d) sheep (e) not enough information is provided to answer the question

Task 6

'X BBTuL Y' means 'X runs faster than Y'. Suppose (1) X2 BBTuL X3, (2) X4 BBTuL X1, and (3) X1 BBTuL X2 what does X1 BBTuL?

(a) Only X2 (b) Only X3 (c) Only X4 (d) X2 and X3

(e) X2 and X4 (f) X3 and X4 (g) X2, X3, and X4

Table 4.5: Example prompt for Task 5 and 6. Task 5 uses the bAbI dataset (Weston et al., 2015); Task 6 uses our own newly created datasets.

Chapter 5

DATASET CREATION

Following the task design ideas and the template finalized in the prompt adjustment, we then create dataset for each task. In each subset of data for Task 1 - Task 4 and Task 6, we create 200 test instances. Each subset for Task 5 contains 250 instances, which are adopted and modified from the bAbI dataset. The creation is detailed in the following sections in this chapter.

5.1 Task 1

For Task 1, a test instance is an $(exp-priorities, val)$ pair, where exp is an arithmetic expression, $priorities$ specify the priorities of the operators in the expression, and val is the correct value of the expression with the specified priorities. During the evaluation, exp and $priorities$ are inserted into the prompt template, and the system output is compared with val to calculate the accuracy. We generate an expression in three steps:

- (1) Pick the number of operators, n , in the expression; in this study we will focus on the cases when n is 2 or 3.
- (2) Randomly sample (with replacement) $n + 1$ operands from all the positive integers with m digits, m ranges from 1 to 5.
- (3) Randomly choose n distinct operators from the four basic operators ('+', '-', '*', and '/') to connect the $n + 1$ operands. We divide the expressions into $2^*5=10$ subsets based on the values of n and m .

To generate priorities of the operators in the expression, when n is 2, the two operators in an expression (e.g., $2+3*4$) follow either the standard priorities (e.g., '*' > '+') or the opposite one (e.g., '+' > '*'). They correspond to the *no-move* and *one-move* scenarios, respectively. When n is 3, there are six possible orders for evaluating the expression. For the simplicity of presenting evaluation

results, we pick three of the six possible orders: one with the standard priorities (i.e. *no move*), one with *one move*, and the last one with *two moves*.

Once we have redefined the priorities, we evaluate the expression based on the priorities to get *val*. We keep an expression in our dataset only if its values differ when the priorities change.¹ For each (n, m) pairs, our dataset consists of 200 expressions with the redefined priorities and corresponding values.

5.2 Task 2

The dataset for Task 2 is derived from the one with Task 1 with two changes. First, for each test instance $(exp-priorities, val)$ in Task 1, we randomly pick three wrong choices of the precedence of the operators in *exp* such that those choices lead to expression values different from *val*. Second, we put the correct and wrong choices in a list, randomly shuffle it, and then assign the choices as Choice A, B, C, and D in a multiple-choice question.

5.3 Task 3

The dataset for Task 3 begins with a manually created seed sentence that contain all six rules in Table 4.2: ‘The yellow dog with a small tail chases the black cat quickly across the park’. We create a test set for Task 3 in the following steps.

- (1) We add the seed sentence to a prompt and ask GPT4 to generate 200 sentences with similar structures.
- (2) We use SpaCy² to parse these sentences to obtain the dependency structure. For sentences that are ambiguous or whose parse trees are incorrect (e.g., with wrong PP attachment), we make minor modifications to the sentences to remove the ambiguities and ensure that the resulting parse trees by SpaCy are correct.³

¹E.g., we will throw away $2+3*1$ because its answer is always 5 regardless of which operator has higher priority.

²<https://spacy.io/>

³For instance, suppose the sentence is ‘NP1 V NP2 PP Adv’, where PP should attach to the verb, but the parser mistakenly attach it to NP2. We just switch the positions of PP and Adv to get the new order ‘NP1 V NP2 Adv PP’, which SpaCy can parse correctly.

- (3) For each (m, n) pairs (where m and n ranges from 0 to 3), we create a dataset with the same 200 test instances; each instance consists of an English sentence, m randomly selected syntactic rules and n randomly selected morphological rules from Table 4.2, and automatically generated gold standard translation by applying syntactic rules to the dependency structure and morphological rules to related words (e.g., M1 to a singular or plural noun).

5.4 Task 4

Task 4’s dataset is built upon Task 3. In task 4, Each test question asks LLMs to infer one word order given two sentence pairs, the English sentence and its Language-A translation. The sentence pairs are selected from different datasets in Task 3 to create the following four scenarios. Suppose the test question is ‘what is the canonical word order in Language-A’. The four scenarios based on the word order in the two Language-A sentences are:

- **1-order:** Both language-A sentences have SOV order, and other word orders (e.g., noun and its modifier) are the same as in English.
- **1-order-noise:** Both language-A sentences have SOV order, but other word orders are different from English (e.g., in Language-A, nouns follow its PP modifier).
- **2-order:** One sentence is in SOV order, and the other is in SVO. Other word orders are the same as in English.
- **2-order-noise:** One sentence is in SOV order, and the other is in SVO; other word orders are different from English (e.g., nouns follow its PP modifier). All the answer choices in Task 4 is fixed. For example, for prompt that asks about canonical order of Lanaguge A, the four answers are fixed as ‘(a) SVO only, (b) VSO only, (c) SVO or SOV, (d) SOV only’.

5.5 Task 5

For Task 5, we adopt the basic deduction test set from bAbI dataset and change the question format from question answering to multiple choice question to make the gold answer unique.

In the original prompt, there are eight statements of two sentence structures: (1) <Person Name> is a <Animal Name>, (2) <Animal Name 1> is afraid of <Animal Name 1>. The question is ‘What is <Person Name> afraid of?’ The gold answer is a type of animal. In order to arrive at the gold answer, two steps of reasoning chain are involved. For instance, for the example displayed in Table 4.5, to answer ‘what is Emily afraid of?’ The two steps - ‘Emily is a cat’ and ‘Cats are afraid of wolves’ - will obtain the gold answer of ‘wolf’. However, from the small-scaled test, GPT4 produces as the final answer ‘Gertrude’, which should also be acceptable as ‘Gertrude is a wolf’ in this example. To make the gold answer unique, we modified the original prompt into multiple question format.

Additional datasets are created by modifying each test instance in the original dataset detailed in Section 4.3. For random string dataset, we replace all the person names and animal names in the prompts with random strings of length four to six, and leave everything unchanged. For negation and quantifier dataset, we add ‘not’ or ‘some’ to the evidence sentences used in the reasoning chain respectively to alter the reasoning chain. For extended dataset, we change the answer choices from animal names to person names so that the deduction requires one more step of ‘<Person Name> is a <Animal Name>’ to arrive at the answer.

5.6 Task 6

For Task 6, we firstly create a template ‘Suppose (1) X1 p X2, (2) X2 p X3, and (3) X4 p X1, what does X1 p’. A test instance is created by replacing the predicate ‘p’ in the template with a random string and assigning the random string one predicate type from Table 4.4. Each predicate type has 200 instances as a separate dataset. For the instances in all four datasets, the answer choices are all the same, namely ‘(a) Only X2, (b) Only X3, (c) Only X4, (d) X2 and X3, (e) X2 and X4, (f) X3 and X4, (g) X2, X3, and X4’. We intentionally keep the answer choices exactly the same across four datasets to eliminate the effect of ordering of answer choices on model performance. At the same time, depending on the predicate type, the gold answer will be different.

Next, we create a comparison set for each predicate type. In the comparison set, we use actual English words rather than random strings as predicate. Instead, the entities in the sentences are replaced by random strings to create 200 instances.

Chapter 6

PROMPT DESIGN

As previously mentioned, we initiated with an original prompt and finalized the prompt for each task by examining the impact of slight modifications in the wording on the performance of LLMs. Additionally, for Task 1, Task 2, and Task 3, we incorporated CoT into the prompts to assist LLMs in avoiding errors identified in the error analysis, aiming to evaluate whether CoT can ultimately enhance model performance. This chapter delineates the design of these CoT prompts.

6.1 Task 1

To test LLMs on Task 1, we experiment with four types of prompts:

- **0-shot**: for each test instance, we simply use the prompt without providing any example or CoT instructions. An example of the full **0-shot** prompt is included here:

*We redefine the priorities of elementary arithmetic operators as follows: '+' = '/'
> '*'.*

*Based on the new priorities, what is the value of $98 * 43 + 29 / 25$? You can give the result as a decimal, and please put the final numeric result in brackets [], for example, [1].*

- **0-shot-CoT**: The prompt includes CoT instructions 'Please do this step by step.' at the end of the prompt, but no demonstration examples are provided in the prompts. An example of the full **0-shot-CoT** prompt is included here:

*We redefine the priorities of elementary arithmetic operators as follows: '+' = '/'
> '*'.*

*Based on the new priorities, what is the value of $98 * 43 + 29 / 25$? Please do this*

step by step. You can give the result as a decimal, and please put the final numeric result in brackets [], for example, [1].

- **1-shot**: the prompt additionally includes an example expression and its correct answer, but no CoT instructions are given to illustrate the solution step by step. An example of the full **1-shot** prompt is included here:

We redefine the priorities of elementary arithmetic operators as follows: '+' = '/' > ''.*

*[Example Question]: What is the value of $62 * 19 + 57 / 74$?*

[Answer]: 63.67567567567567.

*Based on the new priorities, what is the value of $98 * 43 + 29 / 25$? Please do this step by step. You can give the result as a decimal, and please put the final numeric result in brackets [], for example, [1].*

- **1-shot-CoT**: the prompt additionally includes an example expression, the correct answer, and the CoT instructions that demonstrates how the example expression should be evaluated at each step with the newly defined precedence.

We redefine the priorities of elementary arithmetic operators as follows: '+' = '/' > ''.*

*[Example Question]: What is the value of $62 * 19 + 57 / 74$?*

[Answer]: Step 1: Since '+' and '/' have the same priority over '', we evaluate '+' and '/' from left to right. In this case, '+' comes before '/', so we calculate '+' first: $19 + 57 = 76$, so the expression becomes $62 * 76 / 74$.*

Step 2: '/' has priority over '', now we calculate '/': $76 / 74 = 1.027027027027027$, so the expression becomes $62 * 1.027027027027027$.*

*Step 3: Finally, $62 * 1.027027027027027 = 63.67567567567567$, so the final answer is [63.67567567567567].*

*Based on the new priorities, what is the value of $98 * 43 + 29 / 25$? Please do this step by step. You can give the result as a decimal, and please put the final numeric*

result in brackets [], for example, [1].

1-shot-CoT prompt is devised after the comprehensive error analysis on 200 instances of two-move expressions with two digits. We have observed that LLMs frequently adhere to standard precedence rules rather than newly defined ones during calculations. As a result, we opted to design demonstration examples featuring identical operators positioned in the same sequence within the prompt's expression. This aims to guide LLMs in executing consistent calculations. Additionally, LLMs occasionally exhibit tendencies to omit or include one step in the calculation process, or use incorrect operands in calculation. Addressing these issues, the CoT instructions comprise three sequential steps. These steps are devised to systematically guide LLMs through the evaluation process, ensuring the use of correct operands at each stage.

6.2 Task 2

In Task 1, the performance of 0-shot-CoT and 1-shot is worse than that of 0-shot. Given that Task 2 is closely related to Task 1, for Task 2, we experiment with 0-shot (as our baseline) and 1-shot-CoT only. In task 2, LLMs are given an expression and its correct value to choose the correct newly defined precedence out of four choices. To answer the question correct, the most practical approach is to evaluate the expression with the precedence in four choices and pick the precedence that can evaluate the expression to the correct answer. Accordingly, we design three types of prompts for 1-shot-CoT (*Isc*). Here are the explanations and examples for four types of prompts.

- **0-shot**, similar to Task 1, is when we use the prompt as it is without providing any instructions.

An example of **0-shot** looks like this:

*For the expression $8 - 4 * 4 + 9$, the precedence of the operators is redefined. If $8 - 4 * 4 + 9 = 52$, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].*

A. '+' = '-' = '*'

B. '-' = '* > '+'

C. '+' > '* = '-'

D. '-' = '+' = '*'

- **1sc-Simple** asks the LLM to go through four choices one by one to determine which is the correct answer. When inspecting an answer choice, **1sc-Simple** only provides the final evaluated value of the expression with the precedence in this answer choice (eg. using the precedence rule of choice A: '*' = '+' = '-', the expression $3 * 9 + 4 - 9$ evaluates to 22) but does not include instructions on how to evaluate the expression step by step. An example of **1sc-Simple** prompt is included here:

[Example Question]:

*For the expression $3 * 9 + 4 - 9$, the precedence of the operators is redefined. If $3 * 9 + 4 - 9 = 22$, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].*

A. '*' = '+' = '-'

B. '+' = '-' > '*'

C. '-' > '+' > '*'

D. '+' > '* > '-'

[Answer]:

*Using the precedence rule in the correct answer choice, the expression $3 * 9 + 4 - 9$ should evaluate to 22. We test four options one by one.*

Step 1: using the precedence rule of choice A: '' = '+' = '-', the expression $3 * 9 + 4 - 9$ evaluates to 22*

Step 2: using the precedence rule of choice B: '+' = '-' > '', the expression $3 * 9 + 4 - 9$ evaluates to 12*

Step 3: using the precedence rule of choice C: '-' > '+' > '', the expression $3 * 9 + 4 - 9$ evaluates to 12*

Step 4: using the precedence rule of choice D: '+' > ' > '-', the expression $3 * 9$*

$+ 4 - 9$ evaluates to 30

Given above, only with choice A, the expression evaluates to 22, so the correct answer should be [A]

For the expression $8 - 4 * 4 + 9$, the precedence of the operators is redefined. If $8 - 4 * 4 + 9 = 52$, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].

- A. '+' = '-' = '*'
- B. '-' = '* > '+'
- C. '+' > '* = '-'
- D. '-' = '+' = '*'

- **1sc-Partial** adds the instruction for evaluating the expression with the precedence in Choice A, and that instruction is the same as the one used in 1-shot-CoT in Task 1. The prompt then asks the LLM to do the same for other choices. An example of **1sc-Partial** prompt is included here:

[Example Question]:

For the expression $3 * 9 + 4 - 9$, the precedence of the operators is redefined. If $3 * 9 + 4 - 9 = 22$, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].

- A. '* = '+' = '-'
- B. '+' = '-' > '*'
- C. '-' > '+' > '*'
- D. '+' > '* > '-'

[Answer]:

Using the precedence rule in the correct answer choice, the expression $3 * 9 + 4 - 9$

should evaluate to 22. We test four options one by one.

For A: The value of $3 * 9 + 4 - 9$ can be calculated as follows: Based on the precedence rules of this answer choice ' $*$ ' = ' $+$ ' = ' $-$ ', $*$, $+$, and $-$ have the same priority, so we can evaluate the expression from left to right Step1: Firstly, we calculate ' $*$ ': $3 * 9 = 27$, so the expression becomes $27 + 4 - 9$. Step2: Then we calculate ' $+$ ': $27 + 4 = 31$, so the expression becomes $31 - 9$. Step3: Finally, $31 - 9 = 22$, so this answer choice evaluates to 22.

Similarly, we evaluate the expressions in B, C, and D in the same way as we evaluate the expression in A.

For B: using the precedence rule of choice B: ' $+$ ' = ' $-$ ' > ' $*$ ', the expression $3 * 9 + 4 - 9$ evaluates to 12

For C: using the precedence rule of choice C: ' $-$ ' > ' $+$ ' > ' $*$ ', the expression $3 * 9 + 4 - 9$ evaluates to 12

For D: using the precedence rule of choice D: ' $+$ ' > ' $*$ ' > ' $-$ ', the expression $3 * 9 + 4 - 9$ evaluates to 30

Given above, only with choice A, the expression evaluates to 22, so the correct answer should be [A]

For the expression $8 - 4 * 4 + 9$, the precedence of the operators is redefined. If $8 - 4 * 4 + 9 = 52$, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].

A. ' $+$ ' = ' $-$ ' = ' $*$ '

B. ' $-$ ' = ' $*$ ' > ' $+$ '

C. ' $+$ ' > ' $*$ ' = ' $-$ '

D. ' $-$ ' = ' $+$ ' = ' $*$ '

- **1sc-Full** is a thorough version of the second type: it adds the instruction for evaluating the

expression with the precedence in all four answer choices. An example of **1sc-Full** prompt is included here:

[Example Question]:

*For the expression $3 * 9 + 4 - 9$, the precedence of the operators is redefined. If $3 * 9 + 4 - 9 = 22$, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].*

- A. '*' = '+' = '-'
- B. '+' = '-' > '*'
- C. '-' > '+' > '*'
- D. '+' > '* > '-'

[Answer]:

*Using the precedence rule in the correct answer choice, the expression $3 * 9 + 4 - 9$ should evaluate to 22. We test four options one by one.*

*For A: The value of $3 * 9 + 4 - 9$ can be calculated as follows: Based on the precedence rules of this answer choice '*' = '+' = '-', *, +, and - have the same priority, so we can evaluate the expression from left to right Step1: Firstly, we calculate '*': $3 * 9 = 27$, so the expression becomes $27 + 4 - 9$. Step2: Then we calculate '+': $27 + 4 = 31$, so the expression becomes $31 - 9$. Step3: Finally, $31 - 9 = 22$, so this answer choice evaluates to 22.*

*For B: The value of $3 * 9 + 4 - 9$ can be calculated as follows: Based on the precedence rules of this answer choice '+' = '-' > '*', '+' and '-' have the same priority, which is higher than that of '*' Step1: Since '+' and '-' have the same priority over *, we evaluate '+' and '-' from left to right. In this case, '+' comes before '-', so we calculate '+' first: $9 + 4 = 13$, so the expression becomes $3 * 13 - 9$. Step2: '-' has priority over '*'. Now we calculate '-': $13 - 9 = 4$, so the expression becomes $3 * 4$. Step3: Finally, $3 * 4 = 12$, so this answer choice evaluates to 12.*

*For C: The value of $3 * 9 + 4 - 9$ can be calculated as follows: Based on the precedence rules of this answer choice $'-' > '+' > '*'$, $'[]'$ has the highest priority, followed by $'+'$. $'*'$ has the lowest priority. Step1: Since $'-'$ has the highest priority, we calculate $'-'$ first: $4 - 9 = -5$, so the expression becomes $3 * 9 + -5$. Step2: Since $'+'$ has priority over $'*'$, we then calculate $'+'$: $9 + -5 = 4$, so the expression becomes $3 * 4$. Step3: Finally, $3 * 4 = 12$, so this answer choice evaluates to 12.*

*For D: The value of $3 * 9 + 4 - 9$ can be calculated as follows: Based on the precedence rules of this answer choice $'+' > '* > '-'$, $'[]'$ has the highest priority, followed by $'*'$. $'-'$ has the lowest priority. Step1: Since $'+'$ has the highest priority, we calculate $'+'$ first: $9 + 4 = 13$, so the expression becomes $3 * 13 - 9$. Step2: Since $'*'$ has priority over $'-'$, we then calculate $'*'$: $3 * 13 = 39$, so the expression becomes $39 - 9$. Step3: Finally, $39 - 9 = 30$, so this answer choice evaluates to 30.*

Given above, only with choice A, the expression evaluates to 22, so the correct answer should be [A]

*For the expression $8 - 4 * 4 + 9$, the precedence of the operators is redefined. If $8 - 4 * 4 + 9 = 52$, what is the redefined precedence of the operators? Please choose the correct answer from A, B, C, and D, and put the final answer choice in brackets, for example, [B].*

- A. $'+' = '-' = '*'$
- B. $'-' = '* > '+'$
- C. $'+' > '* = '-'$
- D. $'-' = '+' = '*'$

The CoT instructions provided for each answer choices in **1sc-Partial** and **1sc-Full** have the same format and content as the CoT instructions used for Task 1. We add the CoT instructions hoping to

address the same mistakes made by LLMs when evaluating each answer choice.

6.3 Task 3

Like Task 1, we use four types of prompts: 0-shot, 0-shot-CoT, 1-shot and 1-shot-CoT.

- **0-shot** simply uses the prompt as it is without providing any instructions. An example is included here:

The following are the linguistic rules of Language-A.

In Language-A, the canonical word order of sentences follow a Subject-Object-Verb structure.

Example: ‘The cat eats the mouse’ should be ordered as ‘The cat the mouse eats’.

Prepositional phrases that modify a verb remain positioned after the verb.

Similarly, adverbs that modify a verb still maintain their position after the verb.

Prepositional phrases that modify a noun remain positioned after the noun.

In Language-A, the morphological rule for nouns involves indicating a singular noun by adding ‘S_’ as a prefix to its lemma, and indicating a plural noun by adding ‘P_’ as a prefix to its lemma.

Example: ‘These cats in the park’ is expressed as ‘These P_cat in the park’ to represent plural and ‘A cat in the park’ is expressed as ‘A S_cat in the park’ to represent singular.

For anything not explicitly specified, Language-A follows the same word order as English.

Here are the translated pairs of English lemmas into lemmas in Language-A: (the, tgt_the), (across, tgt_across), (field, tgt_field), (a, tgt_a), (farmer, tgt_farmer), (carefully, tgt_carefully), (bright, tgt_bright), (., .), (smile, tgt_smile), (basket, tgt_basket), (carry, tgt_carry), (heavy, tgt_heavy), (with, tgt_with) .

Translate the following simple sentence into Language-A: “The farmer with a bright smile carries the heavy basket carefully across the field .”.

Please put the final translated result in brackets [] at the end of your response, for example, [tgt_the tgt_cat tgt_the tgt_dog tgt_chases].

- **0-shot-CoT** only adds ‘Let’s do this step by step’ to the end of the prompt. An example of this type of prompt is included here:

The following are the linguistic rules of Language-A.

In Language-A, the canonical word order of sentences follow a Subject-Object-Verb structure.

Example: ‘The cat eats the mouse’ should be ordered as ‘The cat the mouse eats’.

Prepositional phrases that modify a verb remain positioned after the verb.

Similarly, adverbs that modify a verb still maintain their position after the verb.

Prepositional phrases that modify a noun remain positioned after the noun.

In Language-A, the morphological rule for nouns involves indicating a singular noun by adding ‘S_’ as a prefix to its lemma, and indicating a plural noun by adding ‘P_’ as a prefix to its lemma.

Example: ‘These cats in the park’ is expressed as ‘These P_cat in the park’ to represent plural and ‘A cat in the park’ is expressed as ‘A S_cat in the park’ to represent singular.

For anything not explicitly specified, Language-A follows the same word order as English.

Here are the translated pairs of English lemmas into lemmas in Language-A: (the, tgt_the), (across, tgt_across), (field, tgt_field), (a, tgt_a), (farmer, tgt_farmer), (carefully, tgt_carefully), (bright, tgt_bright), (., .), (smile, tgt_smile), (basket, tgt_basket), (carry, tgt_carry), (heavy, tgt_heavy), (with, tgt_with) .

Translate the following simple sentence into Language-A: “The farmer with a bright smile carries the heavy basket carefully across the field .”.

Please put the final translated result in brackets [] at the end of your response, for example, [tgt_the tgt_cat tgt_the tgt_dog tgt_chases].

Please think step by step.

- **1-shot** provides LLMs with a demonstration example of translation but no information is given on how to translate the English sentence to Language-A. An example of the full **1-shot** prompt is included here:

The following are the linguistic rules of Language-A.

In Language-A, the canonical word order of sentences follow a Subject-Object-Verb structure.

Example: 'The cat eats the mouse' should be ordered as 'The cat the mouse eats'.

Prepositional phrases that modify a verb remain positioned after the verb.

Similarly, adverbs that modify a verb still maintain their position after the verb.

Prepositional phrases that modify a noun remain positioned after the noun.

In Language-A, the morphological rule for nouns involves indicating a singular noun by adding 'S_' as a prefix to its lemma, and indicating a plural noun by adding 'P_' as a prefix to its lemma.

Example: 'These cats in the park' is expressed as 'These P_cat in the park' to represent plural and 'A cat in the park' is expressed as 'A S_cat in the park' to represent singular. Prepositional phrases that modify a verb remain positioned after the verb.

Adverbs that modify a verb remain positioned after the verb.

For anything not explicitly specified, Language-A follows the same word order as English.

Here's an example that demonstrates how to translate an English sentence into Language-A.

Example sentence: The black cat with a small tail eats the food quietly in the room .

Below, you'll find the translated pairs of English lemmas into lemmas in Language-A for the example sentence: [('the', 'tgt_the'), ('black', 'tgt_black'),

(‘cat’, ‘tgt_cat’), (‘with’, ‘tgt_with’), (‘a’, ‘tgt_a’), (‘small’, ‘tgt_small’), (‘tail’, ‘tgt_tail’), (‘eats’, ‘tgt_eats’), (‘food’, ‘tgt_food’), (‘quietly’, ‘tgt_quietly’), (‘in’, ‘tgt_in’), (‘room’, ‘tgt_room’), (‘.’, ‘.’)] *Translated example sentence: tgt_the tgt_black S_tgt_cat tgt_with tgt_a tgt_small S_tgt_tail tgt_the S_tgt_food tgt_eats tgt_quietly tgt_in tgt_the S_tgt_room .*

Here are the translated pairs of English lemmas into lemmas in Language-A: (the, tgt_the), (across, tgt_across), (field, tgt_field), (a, tgt_a), (farmer, tgt_farmer), (carefully, tgt_carefully), (bright, tgt_bright), (., .), (smile, tgt_smile), (basket, tgt_basket), (carry, tgt_carry), (heavy, tgt_heavy), (with, tgt_with) .

Translate the following simple sentence into Language-A: “The farmer with a bright smile carries the heavy basket carefully across the field .”.

Please put the final translated result in brackets [] at the end of your response, for example, [tgt_the tgt_cat tgt_the tgt_dog tgt_chases].

- **1-shot-CoT**, by contrast, not only transforms sentences but also explicates the reasoning behind each step of the translation, offering a narrative that explains the application of Language-A’s rules. An example of the full **1-shot-CoT** prompt is included here:

The following are the linguistic rules of Language-A.

In Language-A, the canonical word order of sentences follow a Subject-Object-Verb structure.

Example: ‘The cat eats the mouse’ should be ordered as ‘The cat the mouse eats’.

Prepositional phrases that modify a verb remain positioned after the verb.

Similarly, adverbs that modify a verb still maintain their position after the verb.

Prepositional phrases that modify a noun remain positioned after the noun.

In Language-A, the morphological rule for nouns involves indicating a singular noun by adding ‘S_’ as a prefix to its lemma, and indicating a plural noun by adding ‘P_’ as a prefix to its lemma.

Example: ‘These cats in the park’ is expressed as ‘These P_cat in the park’ to

represent plural and 'A cat in the park' is expressed as 'A S_cat in the park' to represent singular. Prepositional phrases that modify a verb remain positioned after the verb.

Adverbs that modify a verb remain positioned after the verb.

For anything not explicitly specified, Language-A follows the same word order as English.

Here's an example that demonstrates how to translate an English sentence into Language-A step by step.

Example sentence: *The black cat with a small tail eats the food quietly in the room .*

Below, you'll find the translated pairs of English lemmas into lemmas in Language-A for the example sentence: [(*'the'*, *'tgt_the'*), (*'black'*, *'tgt_black'*), (*'cat'*, *'tgt_cat'*), (*'with'*, *'tgt_with'*), (*'a'*, *'tgt_a'*), (*'small'*, *'tgt_small'*), (*'tail'*, *'tgt_tail'*), (*'eats'*, *'tgt_eats'*), (*'food'*, *'tgt_food'*), (*'quietly'*, *'tgt_quietly'*), (*'in'*, *'tgt_in'*), (*'room'*, *'tgt_room'*), (*'.'*, *'.'*)]

To change the word order of the sentence to Subject-Object-Verb, we first identify the root verb, which is *'eats'*.

Next, we find the subject of the sentence, *'cat'*, and its related tokens: [*'The'*, *'black'*, *'with'*, *'a'*, *'small'*, *'tail'*]. Then, we locate the object of the sentence, *'food'*, and its related tokens: [*'the'*].

We then move the object and its related tokens behind the subject and its related tokens.

Now, the sentence is in Subject-Object-Verb word order as applied in Language-A: *'The black cat with a small tail the food eats quietly in the room .'*

We translate the English sentence into Language-A using the lemma pairs between English and Language-A:

tgt_the tgt_black tgt_cat tgt_with tgt_a tgt_small tgt_tail tgt_the tgt_food tgt_eat tgt_quietly tgt_in tgt_the tgt_room . To apply the morphological rule for nouns, add *'S_'* as a prefix for singular nouns and *'P_'* as a prefix for plural nouns in their lemmas:

First identify singular nouns: tgt_cat, tgt_tail, tgt_food, tgt_room.

Then add 'S_' as a prefix to each singular noun: S_tgt_cat, S_tgt_tail, S_tgt_food, S_tgt_room

There is no plural noun in the sentence.

Now, the sentence follows the singular and plural noun rule in Language-A: 'tgt_the tgt_black S_tgt_cat tgt_with tgt_a tgt_small S_tgt_tail tgt_the S_tgt_food tgt_eat tgt_quietly tgt_in tgt_the S_tgt_room .'.

Lastly, we have the translated example sentence: tgt_the tgt_black S_tgt_cat tgt_with tgt_a tgt_small S_tgt_tail tgt_the S_tgt_food tgt_eats tgt_quietly tgt_in tgt_the S_tgt_room .

Here are the word pairs between English and the Language-A: Here are the translated pairs of English lemmas into lemmas in Language-A: (the, tgt_the), (across, tgt_across), (field, tgt_field), (a, tgt_a), (farmer, tgt_farmer), (carefully, tgt_carefully), (bright, tgt_bright), (., .), (smile, tgt_smile), (basket, tgt_basket), (carry, tgt_carry), (heavy, tgt_heavy), (with, tgt_with) .

Translate the following simple sentence into Language-A: "The farmer with a bright smile carries the heavy basket carefully across the field .".

Please put the final translated result in brackets [] at the end of your response, for example, [tgt_the tgt_cat tgt_the tgt_dog tgt_chases].

To explain how the CoT instructions in **1-shot-CoT** is designed to target system mistakes, consider the sentence 'The cat chases the mouse.' In the CoT process, this sentence is first analyzed token-by-token. The script then applies Language-A's SOV structure and reorder the sentence as 'The cat the mouse chases.' At the same time, the CoT process provides a running commentary, explaining that the object is moved after the subject and before the verb due to the SOV rule, and the verb 'chases' is marked as the root verb. These steps are created to correct the mistakes made by LLMs where LLMs could not successfully apply the rule to the relevant elements in the sentence. Then each words in the English sentence is mapped to the words in Language-A. This step instructs LLMs to perform the correct order of mapping as they sometimes switch the positions of some words when they perform word by word mapping. If Language-A has morphological rules such as marking

singular and plural nouns or specific tense indicators for verbs, these are applied next. For instance, if ‘cat’ is singular, it might be transformed to ‘S_cat,’ and if the verb ‘chase’ is in the present tense, it could become chase_+. The final step here teaches LLMs to apply morphological rules to the relevant words in the sentence.

6.4 Task 4

In Task 4, we ask LLMs to infer about the rules in Language-A by providing them with translation sentence pairs and word pairs. In this task, we only experiment with the 0-shot setting. An example of the prompt in Task 4 asking about the canonical order is included here:

‘The farmer with a bright smile carries the heavy basket carefully across the field’ in English is translated to ‘tgt_the tgt_farmer tgt_with tgt_a tgt_bright tgt_smile tgt_the tgt_heavy tgt_basket tgt_carries tgt_carefully tgt_across tgt_the tgt_field’ in Language A.

‘The skilled chef in the bustling kitchen prepares the delicious meal quickly on the stove’ in English is translated to ‘tgt_the tgt_skilled tgt_chef tgt_in tgt_the tgt_bustling tgt_kitchen tgt_the tgt_delicious tgt_meal tgt_prepares tgt_quickly tgt_on tgt_the tgt_stove’ in Language A.

In the first sentence, the noun ‘farmer’ is modified by the prepositional phrase ‘with a bright smile’. The verb ‘carries’ is modified by the prepositional phrase ‘across the field’. In the second sentence, the noun ‘chef’ is modified by the prepositional phrase ‘in the bustling kitchen’. The verb ‘prepares’ is modified by the prepositional phrase ‘on the stove’.

Here are the word pairs between English and the language A: (a, tgt_a), (across, tgt_across), (basket, tgt_basket), (bright, tgt_bright), (bustling, tgt_bustling), (carefully, tgt_carefully), (carries, tgt_carries), (chef, tgt_chef), (delicious, tgt_delicious), (farmer, tgt_farmer), (field, tgt_field), (heavy, tgt_heavy), (in, tgt_in), (kitchen, tgt_kitchen), (meal, tgt_meal), (on, tgt_on), (prepares, tgt_prepares), (quickly, tgt_quickly), (skilled, tgt_skilled), (smile, tgt_smile), (stove, tgt_stove), (the, tgt_the), (with, tgt_with).

What can we observe about the canonical order of language A? Your answer should

be based solely on the canonical word order in the given sentence pairs. Do not infer the canonical word order based on word order of other components in the sentences such as the order between nouns and their modifiers; for instance, do not use universal implicatures for inference. You can assume that each pair of sentences has the same syntactic structures, with the exception that word order may vary between children of a node in the syntactic structures. This means there are no translation divergences that alter the syntactic structure. Given this context, please determine the correct syntactic characteristic of Language A:

- A. Language A follows SVO word order*
- B. Language A follows SOV word order*
- C. Language A follows VSO word order*
- D. Language A follows either SVO or SOV word order*

To respond, please provide your reasoning and indicate your final answer in brackets at the end of your response, for example [A].

6.5 Task 5

In Task 5, we modify the basic deduction test set into multiple choice format as our baseline, and then further modify the eight statements or the answer choice to create other subsets of data. For this task, we only experiment with the 0-shot prompt. An example of the prompt modified with negation in Task 5 is presented here:

- (1) Wolves are afraid of mice.*
- (2) Sheep are afraid of mice.*
- (3) Winona is a sheep.*
- (4) Mice are afraid of cats.*
- (5) Cats are afraid of wolves.*
- (6) Jessica is a mouse.*
- (7) Emily is not a cat.*
- (8) Gertrude is a wolf.*

Based on the above sentences, what is Emily afraid of?

- (a) *cat*
- (b) *mouse*
- (c) *wolf*
- (d) *sheep*
- (e) *not enough information is provided to answer the question*

6.6 Task 6

In Task 6, we test LLMs' ability to discern semantic nuances of four types of predicates. Again, for Task 6, the only experiment we have done is for 0-shot prompt. An example of the prompt asking about transitive and symmetrical predicate is included here:

Given the three statements here:

- (1) $X1 \ xzpQY \ X2$
- (2) $X4 \ xzpQY \ X1$
- (3) $X2 \ xzpQY \ X3$

If 'X xzpQY Y' means 'X and Y have the same speed', what does $X1 \ xzpQY$?

- (a) *Only X2*
- (b) *Only X3*
- (c) *Only X4*
- (d) *X2 and X3*
- (e) *X2 and X4*
- (f) *X3 and X4*
- (g) *X2, X3, and X4*

For comparison purposes, we have also created for each of the four predicate types a different set, where we use English words as predicate but random strings to replace entities in the sentence. An example of the prompt asking about transitive and symmetrical predicate for the comparison dataset is included here:

Given the three statements here:

- (1) *jm1 and jm2 have the same speed.*

(2) *jm2 and jm3 have the same speed.*

(3) *jm4 and jm1 have the same speed.*

What and jm1 have the same speed?

(a) *Only jm2*

(b) *Only jm3*

(c) *Only jm4*

(d) *jm2 and jm3*

(e) *jm2 and jm4*

(f) *jm3 and jm4*

(g) *jm2, jm3, and jm4*

Chapter 7

EXPERIMENTS

We run the experiments on GPT4-Turbo¹, LLaMA-2-chat-70B (LLaMA) (Touvron et al., 2023) and Flan-T5-XXL (T5) (Raffel et al., 2020). Notably, the performance of T5 demonstrates significant deficiencies. Its output is often incomplete or erroneous, so we choose not to report T5 results. LLaMA exhibits lower performance compared to GPT4 in almost all tasks and a full performance comparison of the two models is detailed at the end of this chapter. In this study, we have opted to mainly report and discuss the experiments conducted with GPT-4², given its superior performance compared to other models. Similarly, our error analysis also centers on the output generated by GPT-4.

In all the tables in this section that report system performance, each cell reports accuracy on 200 test instances except for Task 5, which uses 250 test instances as in the original bAbI dataset.

For Task 1-3, we also run the experiment on prompts with different levels of CoT as detailed in Section 6 and report the model performance here.

7.1 Task 1: Arithmetic Expression Calculation

For Task 1, we run the experiment for zero move, one move, and two moves settings (as defined in Sect 2.1); each operand has one to five digits. The result is in Table 7.1, which shows that the system performance decreases as the number of digits in the operands increases, consistent with previous findings (Brown et al., 2020; Touvron et al., 2023). The system performance also deteriorates significantly when the precedence is different from the standard one (e.g., the accuracy of two digits calculation for two moves dropped drastically from 95.0% to 39.5%) under the 0-shot setting.

After detailed examination of errors for two moves expressions with two digit operands, we modified the prompt with 1-shot CoT, which greatly improved model performance across all the

¹We use their API at <https://platform.openai.com>

²To make our experiments reproducible, we select the token with the highest probability when GPT4 generates text token by token.

experiments (see Row 2, 4, 6).

The system outputs for one example of four types of prompts are in Appendix A.

7.2 Task 2: Choosing Precedence of Operators

In Task 1, we observed GPT4’s subpar performance with redefined two-move precedence under 0-shot setting. For Task 2, we report GPT4’s performance on two-move expressions with four types of prompts in Table 7.2. The correct answer is uniformly distributed among the four choices, so the accuracy of a random guess would be 25%. Notably, the ‘0-shot’ accuracy is worse than random guess. This is primarily caused by GPT4’s low performance on Task 1 and its tendency of not picking any given choice and hence producing an invalid answer. The combination of low accuracy and a high invalid percentage is expected because Task 1 has shown GPT4’s inability to consistently evaluate expressions with novel precedence rules.

Similar to Task 1, 1-shot-CoT in Task 2 also leads to better performance. Nevertheless, GPT4’s performance is still low and it still produces invalid answers more than half of the time. This result is again unsurprising: although 1-shot-CoT in Task 1 greatly improves accuracy, in Task 2, the models need to evaluate the expression correctly four consecutive times to reliably choose the correct answer. A wrong evaluation in any of the four choices could potentially lead to incorrect or invalid answer.

The system outputs for one example in Task 2 are in Appendix B.

7.3 Task 3: Machine Translation

In Task 3, LLMs are asked to translate some English sentences into Language-A. For evaluation, we use four metrics: exact match (EM), BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and Morph F1-score. EM is the strictest, requiring the system output to be exactly the same as the gold standard; BLEU and METEOR are more lenient, as wrong word order will only affect some word ngrams. Morph F1 evaluates LLMs’ ability to apply morphological rules by calculating precision and recall on words requiring morphological changes. METEOR is expanded to include aligned chunk numbers (C) and unaligned word count (UA) to assess word order and morphological accuracy.

Table 7.3 shows the performance of GPT4 with 0-shot prompts when different numbers of

ID	Move	AP	1D	2D	3D	4D	5D
(a) number of operators = 2							
1		0-s	100	97.0	80.0	50.5	43.5
2	no	0-s-c	99.5	98.5	71.0	51.0	41.5
3	move	1-s	100	99.0	82.5	52.5	46.5
4		1-s-c	100	100	82.0	53.5	47.5
5		0-s	94.5	91.5	80.0	57.0	45.5
6	one	0-s-c	97.0	96.0	82.0	53.0	44.5
7	move	1-s	89.5	93.5	74.0	55.0	43.5
8		1-s-c	100	100	88.5	65.5	57.0
(b) number of operators = 3							
9		0-s	99.0	95.0	67.5	31.5	20.0
10	no	0-s-c	99.5	95.5	66.5	26.5	19.0
11	move	1-s	99.5	97.5	69.0	27.5	18.5
12		1-s-c	100	98.5	69.0	28.0	20.5
13		0-s	75.0	58.5	40.5	17.5	14.5
14	one	0-s-c	74.5	57.0	32.5	11.5	11.0
15	move	1-s	67.0	58.5	36.0	17.5	12.0
16		1-s-c	94.5	97.5	73.0	35.5	32.5
17		0-s	54.0	39.5	32.5	15.5	13.5
18	two	0-s-c	46.0	42.0	24.0	10.0	7.0
19	moves	1-s	41.0	36.0	25.5	13.5	11.5
20		1-s-c	92.0	94.0	79.5	48.5	46.0

Table 7.1: The accuracy of GPT4-Turbo on Task 1. “AP” denotes the approach, where “s” and “c” denote “*shot*” and “*CoT*”, respectively (e.g., “1-s-c” means 1-shot-CoT). Col 1D-5D refers to the numbers of digits in operands, ranging from 1 to 5. Arithmetic expressions used in each column within (a) and (b) are the same, but the priorities of the operators in the expressions are different between no move, one move, and two moves. The system performance decreased with the increase of digits. In some cases, no move expressions have a lower accuracy than one-move or two-move expressions of 1-s-c prompt. This might be caused by the fact that when digit is larger than 3, the system makes many calculation mistakes, but 1-s-c is not designed at correct calculation mistakes.

	1D	2D	3D
0-shot	25.0 / 52.0	21.5 / 61.0	19.0 / 62.0
1sc-Simple	21.0 / 31.0	23.0 / 40.5	16.0 / 39.0
1sc-Partial	30.5 / 61.5	30.5 / 62.0	22.5 / 64.0
1sc-Full	41.5 / 53.0	43.5 / 53.0	30.5 / 64.5

Table 7.2: Performance of GPT4 on Task 2 under two-move configuration. Each cell has the format x / y , where x is the system accuracy and y is the percentage of *invalid* answers (i.e., GPT4 does not pick any choice).

Language-A rules are included in the prompts. When no rules applies (Expt 1), GPT4’s output matches the gold standard perfectly, indicating that it understands the task and is able to translate an English word to its counterpart. The introduction of linguistic rules led to a decline in performance, notably more severe for strict metrics like EM, which plummeted from 100% in the no-rule scenario (S=0 M=0) to 1% when three syntactic and three morphological rules are used to define Language-A. As the number of rules increases, the system performance decreases across all metrics. The decrease is smaller for lenient measures such as BLEU and METEOR Score, but much more dramatic for strict measures such as EM and aligned chunk number C. This comparative result suggests the model is able to apply some rules, not all, in the translation.

Table 7.4 compare the four prompts when the number of syntactic or morphological rules is three. When comparing the impact of syntactic (S=3 M=0) vs. morphological (S=0 M=3) rules, the system’s performance dipped more sharply with morphological complexities, For instance, in the scenario with three morphological rules (c), the 1-shot-CoT prompt saw EM decrease to 49%, and METEOR scores adjust to 95.6%, highlighting that whether the prompt includes CoT or not, handling morphological rules is more challenging for GPT4. Finally, 1-shot-CoT provides a considerable boost as it is the case in Task 1-2.

The system outputs for one example in Task 3 are in Appendix C.

ID	S	M	EM	BLEU	METEOR			F1
					Score	C	UA	
1	0	0	100	100	100	1.00	0.0	-
2	0	1	69.0	98.5	96.3	1.51	0.54	85.9
3	0	2	46.0	97.8	94.0	1.94	0.87	88.3
4	0	3	30.0	95.5	89.5	2.56	1.59	87.9
5	1	0	67.5	97.6	95.7	2.10	0.29	-
6	1	1	25.5	94.8	88.0	2.57	1.46	60.0
7	1	2	10.0	90.6	76.8	3.75	3.11	87.8
8	1	3	4.5	89.1	71.6	4.12	4.22	61.5
9	2	0	61.5	95.4	91.3	2.21	0.93	-
10	2	1	18.0	93.7	84.9	3.59	1.73	59.7
11	2	2	8.0	91.3	77.4	4.27	3.03	60.6
12	2	3	0.5	89.0	69.5	5.00	4.46	58.3
13	3	0	66.0	97.0	94.1	2.08	0.6	-
14	3	1	11.0	93.8	85.4	3.51	1.69	61.5
15	3	2	4.5	91.6	77.3	4.65	2.91	64.2
16	3	3	1.0	88.8	68.0	5.58	4.45	59.3

Table 7.3: The performance of GPT4 on Task 3 with 0-shot prompts. ‘S’ and ‘M’ represent the number of syntactic and morphological rules included in the prompts. ‘EM’ is exact match, ‘F1’ is Morph F1, and for METEOR, we report the METEOR score, the number of aligned chunks C and the number of unaligned target words UA.

	EM	BLEU	METEOR			F1
			Score	C	UA	
(a) S=0 M=0						
0-s	100	100	100	1.00	0.0	-
(b) S=3 M=0						
0-s	66.0	97.0	94.1	2.08	0.6	-
0-s-c	60.0	99.9	98.3	2.81	0	-
1-s	77.0	99.9	99.3	1.90	0.01	-
1-s-c	93.0	99.8	99.8	1.27	0.01	-
(c) S=0 M=3						
0-s	30.0	95.5	89.5	2.56	1.59	87.9
0-s-c	27.0	96.3	90.1	2.33	1.62	86.8
1-s	54.0	98.4	96.9	1.46	0.57	94.3
1-s-c	49.0	98.6	95.6	1.51	0.64	93.1
(d) S=3 M=3						
0-s	1.0	88.8	68.0	5.58	4.45	59.3
0-s-c	1.5	90.8	75.6	5.44	3.86	62.4
1-s	19.5	98.0	92.3	3.45	1.06	90.3
1-s-c	33.5	97.3	93.4	2.10	1.11	88.9

Table 7.4: The performance of GPT4 on Task 3 with four types of prompts

7.4 Task 4: Inferring Word Order

Following observations from Task 3 that GPT4 performed relatively well with syntactic rules than morphological rules, we aimed to further assess its capability to identify syntactic rules in Language-A when given two sentence translation pairs between English and Language-A. Recall there are four scenarios detailed in Section 5.3 based on the word order in the two Language-A sentences: **1-order**, **1-order-noise**, **2-order**, and **2-order-noise**.

For the latter two scenarios, the results show that the order of the two sentence pairs matters much. For instance, for the canonical order rule, in our original experiments, the first sentence is in SOV, and the second is in SVO. If we switch the order of the sentence pairs, GPT4’s performance increases. The improvement is more significant if the test question is about PV or PN rules, as shown in Table 7.5. The table shows that GPT4 achieves high accuracy in the 1-order setting and can handle noise well in the 1-order-noise setting. Under 2-order scenario, instead of choosing the correct option that says both orders are possible, GPT4 often chooses the option of one order only, leading to much worse performance. We will look into this problem in detail in the next chapter.

	Original Exp.			Switch Exp.		
	SOV	PV	PN	SOV	PV	PN
1-order	83.0	69.5	97.5	87.0	78.5	97.5
1-order-noise	95.0	66.0	98.0	91.0	64.5	96.0
2-order	2.5	17.5	14.0	7.0	77.5	88.5
2-order-noise	3.0	17.0	14.5	14.0	62.0	75.5

Table 7.5: GPT4’s accuracy on Task 4. See Section 3.4 for the meaning of the row and column headers.

7.5 Task 5: Basic Deduction

In Task 5, we run experiments on the original bAbI dataset and its modified versions. The accuracy for task 5 is reported in Table 7.6.

GPT4 achieved an accuracy of 90.8% on the original dataset. When we test GPT4 on 250 new test

examples whose length of inference chains is three (increased from two in the original dataset), its performance decreased to 85.6%. Replacing nouns with random names or adding quantifiers actually increase the accuracy a little bit, which could be surprising as those changes should conceptually make the task harder. Upon detailed analysis of errors made on the original test, we have found an explanation, which is discussed in the next chapter.

	Base	Ran	Neg	Quan	Ext
Accuracy	90.80	95.20	98.80	91.20	85.60

Table 7.6: Accuracy of GPT4 on Task 5. ‘Base’: original dataset from bAbI; ‘Ran’: replace nouns with random strings; ‘Neg’: add ‘not’ to the predicate; ‘Quan’: add ‘some’ to the subject of the sentence; ‘ext’: extend the reasoning chain.

7.6 Task 6: Deduction with Different Types of Predicates

	NT & NS	NT & S	T & NS	T & S
Random Predicate	99.00	87.00	79.50	52.00
Random Entity	100.00	100.00	96.00	56.50

Table 7.7: Accuracy of GPT4 on Task 6. ‘T’: transitive predicate; ‘S’: symmetrical predicate. ‘NT’: none-transitive predicate. ‘NS’: none-symmetrical predicate. ‘random predicate’ is when the predicate is replaced by random strings in the statements; ‘random Entity’ is when the entities are replaced by random strings in the statements.

We run two sets of experiments respectively on four types of predicate meaning. The accuracy for Task 6 is reported in Table 7.7. As shown in ‘random predicate’ category: when predicates do not carry transitivity or symmetry, GPT4’s performance is exceptional, reaching 99% accuracy. However, when the predicates carry the semantic meaning of transitivity and symmetry, GPT4 begins to show a weaker performance. While it handles symmetrical predicate meaning relatively well, achieving 87% accuracy, it is comparatively weak at discerning the transitivity in semantics. When the predicate combines both aspects, the accuracy is only 52%.

Replacing the entities, or the nouns, rather than the predicates in the statements with random strings consistently show improvements across four types. This suggests that changing key information into random strings may weaken model behavior.

7.7 Comparison of GPT4 and LLaMA

In this section, we include the full comparison of the performance between GPT4 and LLaMA over the six tasks. Again, the results of T5 are not reported because the system output is either incomplete or erroneous. Please also note Task 2 and Task 3 are not included here as the accuracy of LLaMA on these both tasks is approaching zero. The comparison between GPT4 and LLaMA of Task 1 is detailed in Table 7.8, of Task 4 is detailed in Table 7.11, of Task 5 is detailed in Table 7.9, and of Task 6 is detailed in Table 7.10.

	No CoT		CoT	
	GPT4	LLaMA	GPT4	LLaMA
No move	95.50	18.00	98.50	46.00
One move	58.50	1.50	97.50	31.00
Two moves	39.50	2.00	94.00	31.50

Table 7.8: Accuracy comparison between GPT4 and LLaMA on Task 1 for two-digit expressions

	Base	Ran	Neg	Quan	Ext
GPT4	90.80	95.20	98.80	91.20	85.60
LLaMA	64.40	45.60	25.20	8.40	59.20

Table 7.9: Accuracy comparison between GPT4 and LLaMA on Task 5; column title adopted from Table 7.6

	NT & NS	NT & S	T & NS	T & S
GPT4	99.00	87.00	79.50	52.00
LLaMA	2.00	11.00	14.00	31.00

Table 7.10: Accuracy comparison between GPT4 and LLaMA on Task 6; column title adopted from Table 7.7

	GPT-4						LLaMA					
	Original Exp.			Switch Exp.			Original Exp.			Switch Exp.		
	SOV	PV	PN	SOV	PV	PN	SOV	PV	PN	SOV	PV	PN
1 ord	83.0	69.5	97.5	87.0	78.5	97.5	9.9	5.0	3.7	8.4	6.49	4.8
1 ord-noise	95.0	66.0	98.0	91.0	64.5	96.0	0.0	6.3	7.2	0.5	5.5	10.3
2 ord	2.5	17.5	14.0	7.0	77.5	88.5	78.0	22.3	26.3	81.2	20.9	20.5
2 ord-noise	3.0	17.0	14.5	14.0	62.0	75.5	81.1	21.9	30.4	79.2	21.4	30.8

Table 7.11: Comparative analysis of GPT-4 and LLaMA performance on Task 4 for original and switch experiments

Chapter 8

ERROR ANALYSIS

We have conducted extensive error analysis across all tasks. In this chapter, we synthesize the findings from the error analysis to answer our four research questions.

8.1 Q1: How effectively can LLMs adapt to new tasks?

Through detailed error analysis of Task 1, 4, and 5, we have discovered that GPT4 is indeed prone to errors when the prompt contains information in conflict with prior knowledge. In Task 1, for two-move expression with two-digit operands, without CoT, GPT4 made 73 mistakes out of 200 instances in which it still uses the standard precedence despite given the newly defined precedence.

Likewise, in Task 4, many translations in Language-A in the prompt contain syntactic rules different from English. Even when GPT4 correctly replicated such translations from the prompt, it still claims Language-A has the same word order as in English. From Table 8.6, we can observe that there are 35 cases of this error type out of 200 for the rule ‘prepositional phrase before noun’ in Language-A.

	Factual	Counterfactual
Correct	11	8
Incorrect	0	9

Table 8.1: Raw count of correct and incorrect predictions made by GPT4 on the original dataset for Task 5 when one evidence statement is factual vs. counterfactual.

In Task 5, we have found that the errors on the original dataset are correlated with the counterfactual condition in the input statements. For instance, the factual statement ‘Mice are afraid of Cats’ is used as the evidence in the reasoning chain 11 times; its counterfactual counterpart ‘Cats are afraid of mice’ is used 17 times. The accuracy count is shown in Table 8.1. Fisher’s exact test returns a

P-value of 0.004, implying that counterfactual statements, namely ‘Cats are afraid mice’, often lead to errors when used as reasoning evidence. This factor can also partially explain why the modified tests have slightly higher accuracy than the original test. The incorporation of random string, the negator ‘not’, or the quantifier ‘some’ will negate the consequence of counterfactuality to various extent, thus improve the accuracy.

Finally, a comparison between Table 7.6 and 7.7 shows LLMs are good at understanding negation and quantifier, but much worse at discerning the semantic nuances introduced by transitivity and symmetry, which are concepts that might not have been learned well by LLMs.

To conclude, the examination of errors made by the system across different tasks reveals that models have a tendency to ignore or miscomprehend information in the prompt that is in conflict with the prior knowledge they have seen during pre-training.

8.2 Q2: How robust are LLMs to minor variations of prompts?

Based on our experiments, LLMs are robust in handling some types of noise in the prompt. As evidenced by Table 7.5 from Task 4, the presence of noise in prompts has a minimal impact on LLMs’ abilities. When targeted rules being asked stay the same, applying other irrelevant rules into the translation pairs or not does not significantly change the model performance.

	SOV	SVO	VSO	SVO&SOV
SOV	0	0	0	0
SVO	0	0	0	0
VSO	0	0	0	0
SVO&SOV	155	39	0	6

Table 8.2: Confusion Matrix for Task 4’s 2-Order Experiment Focused on SOV Rule. Rows represent the correct answers, while columns indicate GPT4’s predictions.

However, they are not robust but rather sensitive to the positioning of the key information in the prompt. For instance, models’ performance increased dramatically when the two sentence pairs are switched in Task 4. Table 8.2 shows the confusion matrix for the 2-order setting when the test

	SOV	SVO	VSO	SVO&SOV
SOV	0	0	0	0
SVO	0	0	0	0
VSO	0	0	0	0
SVO&SOV	37	135	0	28

Table 8.3: Confusion Matrix for Task 4’s 2-Order Switch Experiment asking about canonical word order. Rows represent the correct answers, while columns indicate GPT4’s predictions.

question asks about canonical word order and the first language-A sentence is in SOV order. Table 8.3 is the confusion matrix when the order of the two sentence pairs are switched. The contrast between the two tables show how sensitive GPT4 is to the position of certain input.

	SOV	SVO	SVO&SOV	SubTotal
SOV 1st	155	39	6	200
SVO 1st	37	135	28	200
Total	192	174	34	400

Table 8.4: Frequency table of SOV Rule Recognition Before and After Sentence Pair Reordering in Tables 8.2 and 8.3. ‘SOV 1st’ indicates scenarios where the first sentence pair follows the SOV structure, and conversely for the switched condition. P-value is less than 0.00001.

This observation was validated through a chi-square analysis detailed in Table 8.4, which measured the impact of sentence pair sequencing on the LLM’s predictive accuracy. The results confirmed a pronounced correlation, emphasizing the significant role that the sequence of presentation plays in the LLM’s capacity to correctly identify and apply syntactic rules.

Similarly, the error analysis of Task 5 reveals that errors are correlated with the position of the reasoning evidence in the given eight statements. Among the eight statements in the prompt, if the first statement is part of the reasoning chain, GPT4 is more likely to make mistakes. To be more specific, out of 23 errors, 13 are caused by the model’s failure to notice and understand the

first statement in the prompt. In the total 250 instances, 93 involve the first statement in the eight statements in the prompt as reasoning evidence, and the rest 157 do not. The error rate with regard to the smallest position of the evidence sentence is reported in Table 8.5.

Both analyses show that the different placements of key information in the prompt will affect model performance.

	Incorrect	Total	Error Rate
1st stmt	13	93	14.0%
Rest stmts	10	157	6.4%

Table 8.5: The count of correct and incorrect instances out of 250 with regard to the position of the evidence in the prompt

8.3 Q3: What are the common error types made by LLMs in our designed tasks?

We have conducted comprehensive error analysis on Task 1 and Task 4.

In Task 1, Errors for two-move expressions with two digits without CoT can be classified into following types¹: (1) false precedence (FP): the model succeeds in translating the newly defined precedence into words but still uses the traditional standard in the subsequent calculations; (2) false interpretation (FI): the model fail to translate the newly defined precedence into words or fail to understand the expression correctly; (3) wrong operands (WO): the model fails to evaluate an operator with operands next to it; (4) missing step (MS): the model fails to execute one step of calculation due to ignorance of the operator in the expression; (5) extra step (ES): the model adds one more step of calculation. The raw count of these error types are detailed in 8.1².

In Task 4, our error analysis, as detailed in Table 8.6, identifies two primary error types made by LLMs. The first type involves misremembering the translated sentence from the prompts, labeled as MR errors. Here, LLMs inaccurately replicate the sentence in Language-A, either altering the word order of the first translated sentence (MR1) or modifying the second translated sentence (MR2). An

¹We omit the error types, such as miscalculation, that have count of one.

²We count multiple errors in one instance separately.

	MR1	MR2	FO1	FO2
original	38	92	35	7
switch	17	2	1	3

Table 8.6: Error Type Distribution for ‘PP Before Noun’ Rule in Task 4: Analyzing 200 Instances. ‘MR’ denotes misremembered translations, with MR1 and MR2 indicating errors in the first and second sentence pairs, respectively. ‘FO’ signifies incorrect observations, distinguishing between the first (FO1) and second (FO2) translated sentences.

example of this error is when the original prompt sentence “the dog with a tail” is incorrectly echoed by the LLM as “with a tail the dog.”

The second type of error, referred to as FO errors, occurs when LLMs correctly copy the translated sentence but then make incorrect observations about it. These errors are divided into two categories: faulty observations about the first translated sentence (FO1) or the second translated sentence (FO2). For example, despite accurately repeating “the dog with a tail” from the prompts, the LLM might erroneously analyze this structure as prepositional phrase before noun.

8.4 Q4: What strategies can be employed to enhance LLM performance on our tasks?

Drawing on the previously mentioned error types and the acknowledged effectiveness of CoT, a promising strategy to boost the performance of LLMs is to craft prompts tailored to guiding models in avoiding those common errors.

After concluding the error types for Task 1, we design two versions of CoT prompt, whose effect is shown in Fig 8.1. The first version of CoT (CoT1) prompt that instructs the model to interpret the new precedence and evaluate in steps, but the positioning of operators in the question expression is different from that in the demonstration example. CoT1 greatly reduced errors of FI, MS, and ES, but not FP. We then design CoT2 to ensure that this positioning is also aligned. The improved CoT drastically reduces errors caused by FP, and eliminates other error types.

In a similar fashion, we have discovered that the errors are correlated with counterfactuality and position of the evidence in the eight statements. In the original bAbI dataset, there are six cases in which the reasoning chain to arrive at the correct answer involves the first statement among the eight statements in the prompt as the evidence, GPT4 answered five of them wrong. Subsequently,

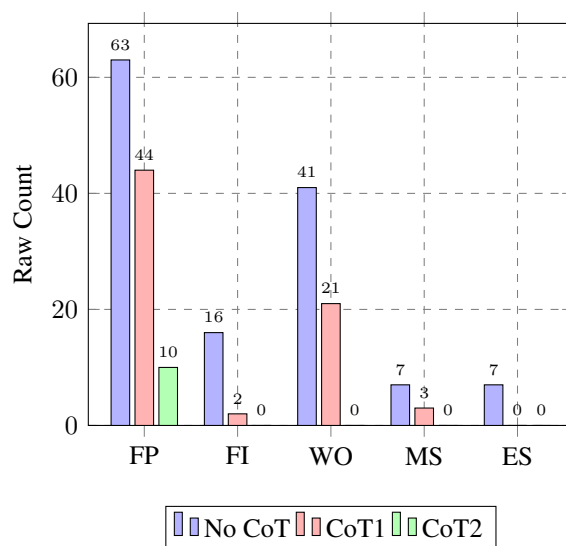


Figure 8.1: Raw count with regard to error types across No CoT, CoT1, and CoT2 conditions for Task 1. The explanation of five types is in Section 8.3

we extract these five instances and add the following sentence to the prompt: ‘The sentences might contain information against common sense; if this happens, stick with the sentences here rather than common sense. Also pay attention to the first sentence.’ GPT4 then answered all five of them correctly.

In conclusion, improving model performance can be achieved through prompt engineering, with CoT structions being a widely embraced method in this domain. However, designing CoT prompts solely from a human perspective may not optimize model performance. Despite its enhancement in performance, the instruction might still overlook system errors. Instead, we advocate for CoT’s optimal effectiveness when it is crafted based on the outcomes of thorough error analysis, specifically targeting and addressing prevalent error patterns discovered in the error analysis.

Chapter 9

DISCUSSION

In this study, we introduced six novel and straightforward tasks aimed at evaluating the performance of Large Language Models (LLMs) on unconventional inputs. While our experiments and error analyses have provided valuable insights into the versatility of LLMs, there remains significant potential for further experiments on a larger scale. In this chapter, we will address the limitations of our study and provide suggestions for future researchers on the evaluations of LLMs.

9.1 *Limitations*

We acknowledge that the tasks we've undertaken only cover a small portion of abilities in the field of LLMs evaluation. Additional tasks are essential to delve into other facets of LLM abilities and to gauge their capacity for generalization to unseen questions. This exploration is crucial in determining whether LLMs are adequately prepared for practical applications in dynamic real-world scenarios. Similarly, although our current study centers on analyzing the output of GPT4, conducting detailed analyses of system outputs from other LLMs can be beneficial. Through thorough error analyses on outputs from various models, common error patterns across different LLMs can be uncovered. This method might hold promise in providing valuable insights into the underlying architectures of LLMs and can contribute significantly to their ongoing development.

Moreover, despite our comprehensive categorization of errors in some tasks, there remains ambiguity regarding the underlying reasons for certain errors made by LLMs. For instance, in Task 4, when the model repeats the wrong translation, it is challenging to know whether the error is a failure in copying or if the model has undergone incorrect syntactic parsing processes. To resolve the uncertainty in this and other error types, it is necessary to devise prompts tailored to discerning the root causes of errors.

9.2 Suggestions on evaluating LLMs

Finally, we humbly offer some advice regarding the evaluation of LLMs. Firstly, it is advisable to avoid bundling together tasks that require disparate skills, as low performance in such cases inspires limited insights into the specific skills that LLMs lack. Instead, the evaluation of LLMs could potentially benefit from simpler and more straightforward tasks that consistently demand the same skills. This approach allows for a focused analysis of those particular skills during error analysis.

Secondly, while it's beneficial to present LLMs with novel tasks, designers should ensure these tasks have reasonable real-life applications. For instance, in Task 3 and Task 4 - machine translation for an artificial language - we simulate the challenge of translating between English and low-resource languages. Given the scarcity of data in such languages, training a model with a vast dataset capable of mapping an input English sentence to the target language is unfeasible. Instead, traditional approaches such as consulting dictionaries and adhering to grammar rules in that language may still be necessary.

Thirdly, the mindset of error analysis and trouble-shooting should apply to every work in the field. In an era where LLMs are rapidly shaping our society, it is crucial for individuals in the community to move beyond simply reporting numerical outcomes and instead focus heavily on investigating the underlying strength and deficiencies that lead to these numbers. By collectively delving into the shortcomings of LLMs, we can together build better LLMs that meet the evolving needs of our society.

Chapter 10

CONCLUSION

In this study, we have developed six new tasks and assessed LLMs performance on these six tasks. The contribution of this thesis is the following. Firstly, we have successfully answered the four research questions raised in the introduction. To summarize: LLMs struggle even with basic tasks when instructions conflict with prior knowledge. Our study also highlights that LLMs display robustness to some types of noise but vulnerability to small variations, such as the different orders of key information and CoT instructions with small difference. Additionally, we have shown that extensive error analysis can guide the design of optimal CoT aimed at eliminating common errors and improving model performance. Furthermore, we have not only implemented but also intend to publicize the scripts used for generating and evaluating datasets for each task. This initiative enables other researchers to replicate our experiments using the current datasets and conduct an even broader error analysis, thereby delving deeper into the limitations of LLMs. Moreover, considering the sensitivity of LLMs to slight variations in prompts and tasks, researchers can leverage these datasets to create new ones and conduct experiments aimed at exploring the versatility of LLMs.

In our future endeavors, we aim to extend our experiments to encompass a broader range of LLMs, such as an upcoming iteration of GPT4. As previously discussed, conducting a comparative error analysis on outputs generated by different LLMs holds the potential to discover common deficiencies among these models. This comparative approach not only enhances our understanding of LLMs but also fosters the ongoing refinement and advancement of these systems. Furthermore, our error analyses have revealed some mistakes made by LLMs could be explained by multiple theories. We intend to devise additional experiments specifically tailored to pinpoint the underlying reasons behind such errors. By systematically dissecting and analyzing these discrepancies, we seek to gain deeper insights into the inner workings of LLMs and thus raise strategies for further improvement.

BIBLIOGRAPHY

- Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 Technical Report. *arXiv preprint arXiv:2305.10403*.
- Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan.
- Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Love-
nia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. <http://arxiv.org/abs/2302.04023> A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity.
- Rachel Bawden and François Yvon. 2023. Investigating the translation performance of a large multilingual language model: the case of bloom. *arXiv preprint arXiv:2303.01911*.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. 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.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-shot Learners. *Advances in neural information processing systems*, 33:1877–1901.
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. <http://arxiv.org/abs/2303.12712> Sparks of artificial general intelligence: Early experiments with gpt-4.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think You Have Solved Question Answering? Try ARC, The AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021a. <http://arxiv.org/abs/2110.14168> Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021b. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. <http://arxiv.org/abs/2104.08758> Documenting large webtext corpora: A case study on the colossal clean crawled corpus.

Simon Frieder, Luca Pinchetti, Alexis Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, and Julius Berner. 2023. <http://arxiv.org/abs/2301.13867> Mathematical capabilities of chatgpt.

Gaël Gendron, Qiming Bao, Michael Witbrock, and Gillian Dobbie. 2024. <http://arxiv.org/abs/2305.19555> Large language models are not strong abstract reasoners.

Shahriar Golchin and Mihai Surdeanu. 2023. <http://arxiv.org/abs/2308.08493> Time travel in llms: Tracing data contamination in large language models.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021a. Measuring Mathematical Problem Solving with the Math Dataset. *arXiv preprint arXiv:2103.03874*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. <http://arxiv.org/abs/2103.03874> Measuring mathematical problem solving with the MATH dataset. *CoRR*, abs/2103.03874.

- Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. <http://arxiv.org/abs/2305.08845> Large language models are zero-shot rankers for recommender systems.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Xiangji Huang. 2023. <http://arxiv.org/abs/2305.18486> A systematic study and comprehensive evaluation of chatgpt on benchmark datasets.
- Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. <http://arxiv.org/abs/2304.03439> Evaluating the logical reasoning ability of chatgpt and gpt-4.
- Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. <http://arxiv.org/abs/2007.08124> Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *CoRR*, abs/2007.08124.
- Inbal Magar and Roy Schwartz. 2022. <https://api.semanticscholar.org/CorpusID:247475929> Data contamination: From memorization to exploitation. *ArXiv*, abs/2203.08242.
- Sewon Min, Xinxu Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. <http://arxiv.org/abs/2202.12837> Rethinking the role of demonstrations: What makes in-context learning work?
- Yonatan Oren, Nicole Meister, Niladri Chatterji, Faisal Ladhak, and Tatsunori B Hashimoto. 2023. Proving Test Set Contamination in Black Box Language Models. *arXiv preprint arXiv:2310.17623*.

- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.
- Pouya Pezeshkpour and Estevam Hruschka. 2023. <http://arxiv.org/abs/2308.11483> Large language models sensitivity to the order of options in multiple-choice questions.
- Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. <http://arxiv.org/abs/2302.06476> Is chatgpt a general-purpose natural language processing task solver?
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. <http://arxiv.org/abs/2103.00020> Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. <http://arxiv.org/abs/1910.10683> Exploring the limits of transfer learning with a unified text-to-text transformer.
- Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A Conversational Question Answering Challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.
- Abulhair Saparov and He He. 2023. <http://arxiv.org/abs/2210.01240> Language models are greedy reasoners: A systematic formal analysis of chain-of-thought.
- Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language Models are Multilingual Chain-of-Thought Reasoners. *arXiv preprint arXiv:2210.03057*.

- Hiroto Tamura, Toshio Hirasawa, Hwicheon Kim, and Mamoru Komachi. 2023. Does Masked Language Model Pre-training with Artificial Data Improve Low-resource Neural Machine Translation? In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2216–2225, Dubrovnik, Croatia.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open Foundation and Fine-tuned Chat Models. *arXiv preprint arXiv:2307.09288*.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. <http://arxiv.org/abs/1905.00537> SuperGlue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. <http://arxiv.org/abs/1804.07461> GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461.
- Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023. <http://arxiv.org/abs/2212.10001> Towards understanding chain-of-thought prompting: An empirical study of what matters.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, 35:24824–24837.
- Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. <https://api.semanticscholar.org/CorpusID:3178759> Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv: Artificial Intelligence*.
- Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2023. <http://arxiv.org/abs/2307.02477> Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks.
- Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2023.

<http://arxiv.org/abs/2306.09841> Are large language models really good logical reasoners? a comprehensive evaluation and beyond.

Xuan Zhang, Navid Rajabi, Kevin Duh, and Philipp Koehn. 2023. Machine Translation with Large Language Models: Prompting, Few-shot Learning, and Fine-tuning with QLoRA. In *Proceedings of the Eighth Conference on Machine Translation*, pages 468–481.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. <http://arxiv.org/abs/2102.09690> Calibrate before use: Improving few-shot performance of language models.

Appendix A

GPT4 OUTPUTS ON FOUR TYPES OF PROMPTS IN TASK 1

In this appendix, we show the full answers produced by GPT4 on an example used in Task 1, where the precedence is '+' = '*' > '-', and the test instance is $2 - 7 + 2 * 5$.

A.1 0-shot output

Based on the new priorities where '+' has the same priority as '', and both are higher than '-', we would perform the operations in the following order:*

*1. Perform the multiplication and addition first (since they have the same priority, we'll perform them from left to right): $2 - 7 + (2 * 5)$*

2. Calculate the multiplication: $2 - 7 + 10$

3. Perform the addition: $2 + 10 = 12$

4. Finally, perform the subtraction: $12 - 7 = 5$

*So, the value of the expression $2 - 7 + 2 * 5$ with the given priorities is [5].*

A.2 0-shot-CoT output

Based on the new priorities you've given, the '+' operator (addition) has the same priority as the '' operator (multiplication), and both are higher than the '-' operator (subtraction). So we need to perform the addition and multiplication before the subtraction. Here's the expression with the new priorities:*

$$2 - 7 + 2 * 5$$

First, we perform the addition and multiplication since they have the same priority. We can do them from left to right:

$$7 + 2 = 9$$

$$2 * 5 = 10$$

Now we replace the original numbers with the results of these operations:

$$2 - 9 * 10$$

Next, we perform the multiplication, which has the same priority as addition but comes next in the sequence:

$$9 * 10 = 90$$

Now we replace the multiplication with its result:

$$2 - 90$$

Finally, we perform the subtraction:

$$2 - 90 = -88$$

So the value of the expression $2 - 7 + 2 * 5$ with the new priorities is [-88].

A.3 1-shot output

Based on the new priorities where '+' has the same priority as '*' and both are higher than '-', we should perform the addition and multiplication before the subtraction.

So for the expression $2 - 7 + 2 * 5$, we first do the addition and multiplication:

$$7 + 2 * 5 = 7 + (2 * 5) = 7 + 10 = 17$$

Now we perform the subtraction:

$$2 - 17 = -15$$

Therefore, the value of the expression $2 - 7 + 2 * 5$ is [-15].

A.4 1-shot-CoT output

Based on the new priorities where '+' and '*' have the same priority and are higher than '-', we evaluate '+' and '*' before '-'. We also evaluate from left to right since '+' and '*' have the same priority.

Step 1: Evaluate '+' before '*'. So we calculate $7 + 2$ first: $7 + 2 = 9$, so the expression becomes $2 - 9 * 5$.

Step 2: Now we evaluate '', which has the same priority as '+', but since there's no '+' left, we just perform the multiplication: $9 * 5 = 45$, so the expression becomes $2 - 45$.*

Step 3: Finally, we perform the subtraction: $2 - 45 = -43$.

So the final answer is [-43].

Appendix B

GPT4 OUTPUTS ON FOUR TYPES OF PROMPTS IN TASK 2

In this section, we show the output of GPT4 with four types of prompts when the test expression $1+9*9/6$ has value 13.666, and the four choices are *<Choice A>*: $'/' > '+' = '*'$, *<Choice B>*: $'*' > '+' = '/'$, *<Choice C>*: $'+' = '*' > '/'$, and *<Choice D>*: $'/' > '+' > '*'$, where *B* is the correct answer, as shown in Table ??.

B.1 0-shot output

*To determine the redefined precedence of the operators that results in the expression $1 + 9 * 9 / 6$ equaling 13.666666666666666, we need to evaluate the expression according to each of the given precedence rules and see which one gives us the correct result. Let's evaluate the expression under each of the given precedence rules:*

A. $'/' > '+' = '*'$

According to this rule, division has the highest precedence, followed by addition and multiplication, which have equal precedence. This means we would perform the division first, then the multiplication, and finally the addition. However, since '+' and '' are of equal precedence, we would normally evaluate them from left to right. But since '/' has higher precedence, we would do the division first:*

$$1 + 9 * (9 / 6)$$

$$1 + 9 * 1.5$$

$$1 + 13.5$$

$$14.5$$

This does not give us the correct result.

B. $'*' > '+' = '/'$

According to this rule, multiplication has the highest precedence, followed by addition and division, which have equal precedence. This means we would perform the

multiplication first, then the addition and division from left to right:

$$(1 + 9) * (9 / 6)$$

$$10 * 1.5$$

$$15$$

This does not give us the correct result either.

$$C. '+' = '*' > '/'$$

According to this rule, addition and multiplication have equal precedence and are higher than division. This means we would perform the addition and multiplication from left to right before the division:

$$(1 + 9) * 9 / 6$$

$$10 * 9 / 6$$

$$90 / 6$$

$$15$$

This also does not give us the correct result.

$$D. '/' > '+' > '*'$$

According to this rule, division has the highest precedence, followed by addition, and then multiplication. This means we would perform the division first, then the addition, and finally the multiplication:

$$1 + (9 * (9 / 6))$$

$$1 + (9 * 1.5)$$

$$1 + 13.5$$

$$14.5$$

This also does not give us the correct result. None of the given precedence rules A, B, C, or D result in the expression equaling 13.666666666666666. However, let's try to find the correct precedence that would give us the desired result:

To get 13.666666666666666, we can try to evaluate the expression by performing the multiplication first, then the division, and finally the addition:

$$9 * 9 = 81$$

$$81 / 6 = 13.5$$

$$1 + 13.5 = 14.5$$

This does not give us the correct result either. Let's try another order: If we perform the division first, then the multiplication, and finally the addition:

$$9 / 6 = 1.5$$

$$1.5 * 9 = 13.5$$

$$1 + 13.5 = 14.5$$

This also does not give us the correct result. It seems there might be an error in the question or the answer choices provided, as none of the precedence rules or alternative calculations result in the expression equaling 13.666666666666666.

If we strictly follow the standard order of operations (PEMDAS/BODMAS), which is parentheses, exponents, multiplication and division (from left to right), and addition and subtraction (from left to right), we would calculate the expression as follows:

$$1 + 9 * 9 / 6$$

$$1 + 81 / 6$$

$$1 + 13.5$$

$$14.5$$

This also does not match the given result of 13.666666666666666. Therefore, based on the information provided, it is not possible to determine the redefined precedence of the operators that would result in the given expression equaling 13.666666666666666. There may be a mistake in the question or the provided answer choices.

B.2 1-shot-CoT-simple output

*To find the correct redefined precedence of the operators for the expression $1 + 9 * 9 / 6$ that evaluates to 13.666666666666666, we need to test each option to see which one results in the given value.*

Step 1: using the precedence rule of choice A: $'/' > '+' = ''$, the expression $1 + 9 * 9 / 6$ evaluates to $1 + 81 / 6$, which is $1 + 13.5 = 14.5$, not 13.666666666666666.*

Step 2: using the precedence rule of choice B: $'' > '+' = '/'$, the expression 1*

$+ 9 * 9 / 6$ evaluates to $1 + (9 * 9) / 6$, which is $1 + 81 / 6 = 1 + 13.5 = 14.5$, not 13.666666666666666.

Step 3: using the precedence rule of choice C: $'+' = '*' > '/'$, the expression $1 + 9 * 9 / 6$ evaluates to $(1 + 9) * (9 / 6)$, which is $10 * 1.5 = 15$, not 13.666666666666666.

Step 4: using the precedence rule of choice D: $'/' > '+' > '*'$, the expression $1 + 9 * 9 / 6$ evaluates to $1 + (9 * (9 / 6))$, which is $1 + (9 * 1.5) = 1 + 13.5 = 14.5$, not 13.666666666666666.

None of the given options A, B, C, or D result in the value 13.666666666666666 when applied to the expression $1 + 9 * 9 / 6$. However, if we consider the original precedence rules without redefining them, where $'*' > '/' > '+'$ and are evaluated before $'+' > '*' > '/'$, the expression evaluates as follows:

$$1 + (9 * 9) / 6 = 1 + 81 / 6 = 1 + 13.5 = 14.5$$

Since none of the options match and the original precedence does not yield the correct result either, there seems to be an error in the question or the provided options. If the expression is supposed to evaluate to 13.666666666666666, then the correct precedence would need to ensure that the division $'/'$ is performed before the multiplication $'*' > '/' > '+'$, which is not reflected in any of the given choices.

B.3 1-shot-CoT-partial output

To find the correct precedence that results in the expression $1 + 9 * 9 / 6$ evaluating to 13.666666666666666, we need to test each of the given options:

For A: Using the precedence rule of choice A: $'/' > '+' = '*'$, the expression $1 + 9 * 9 / 6$ can be calculated as follows: Step1: Since $'/'$ has the highest priority, we calculate $'/'$ first: $9 / 6 = 1.5$, so the expression becomes $1 + 9 * 1.5$. Step2: Since $'+' > '*' > '/'$ have the same priority, we calculate from left to right: $1 + 9 = 10$, so the expression becomes $10 * 1.5$. Step3: Finally, $10 * 1.5 = 15.0$, so this answer choice evaluates to 15.0.

For B: Using the precedence rule of choice B: $'*' > '+' = '/'$, the expression $1 + 9 * 9 / 6$ can be calculated as follows: Step1: Since $'*' > '+' = '/'$ has the highest priority, we calculate

'*' first: $9 * 9 = 81$, so the expression becomes $1 + 81 / 6$. Step2: Since '+' and '/' have the same priority, we calculate from left to right: $81 / 6 = 13.5$, so the expression becomes $1 + 13.5$. Step3: Finally, $1 + 13.5 = 14.5$, so this answer choice evaluates to 14.5.

For C: Using the precedence rule of choice C: '+' = '*' > '/', the expression $1 + 9 * 9 / 6$ can be calculated as follows: Step1: Since '+' and '*' have the same priority and are higher than '/', we calculate from left to right: $1 + 9 = 10$, so the expression becomes $10 * 9 / 6$. Step2: Then we calculate '*': $10 * 9 = 90$, so the expression becomes $90 / 6$. Step3: Finally, $90 / 6 = 15.0$, so this answer choice evaluates to 15.0.

For D: Using the precedence rule of choice D: '/' > '+' > '*', the expression $1 + 9 * 9 / 6$ can be calculated as follows: Step1: Since '/' has the highest priority, we calculate '/' first: $9 / 6 = 1.5$, so the expression becomes $1 + 9 * 1.5$. Step2: Since '+' has the next highest priority, we calculate '+': $1 + 9 = 10$, so the expression becomes $10 * 1.5$. Step3: Finally, $10 * 1.5 = 15.0$, so this answer choice evaluates to 15.0.

None of the options A, B, C, or D result in the expression evaluating to 13.666666666666666. However, if we consider the correct result, we can deduce the correct precedence:

$$1 + 9 * 9 / 6 = 13.666666666666666$$

To achieve this, the multiplication must happen first, followed by the division, and then the addition:

$$\text{Step1: } 9 * 9 = 81$$

$$\text{Step2: } 81 / 6 = 13.5$$

$$\text{Step3: } 1 + 13.5 = 14.5$$

This result does not match the given value of 13.666666666666666. It seems there might be a mistake in the question or the options provided. If we assume the division should happen before the multiplication, we get:

$$\text{Step1: } 9 / 6 = 1.5$$

$$\text{Step2: } 1.5 * 9 = 13.5$$

$$\text{Step3: } 1 + 13.5 = 14.5$$

Again, this does not match the given value. However, if we consider the division and

multiplication to have the same precedence and perform the operations from left to right, we get:

$$\text{Step1: } 9 * 9 = 81$$

$$\text{Step2: } 81 / 6 = 13.5$$

$$\text{Step3: } 1 + 13.5 = 14.5$$

This still does not match the given value. It seems there is an error in the question or the options provided, as none of the precedence rules result in the expression evaluating to 13.666666666666666. If we were to calculate the expression with standard operator precedence (multiplication and division before addition), we would get:

$$9 * 9 = 81$$

$$81 / 6 = 13.5$$

$$1 + 13.5 = 14.5$$

This suggests that the given value of 13.666666666666666 may be incorrect, or the question may be missing the correct precedence option.

B.4 1-shot-CoT-full output

To find the correct precedence that makes the expression $1 + 9 * 9 / 6$ equal to 13.666666666666666, we need to test each of the given options:

For A: The value of $1 + 9 * 9 / 6$ can be calculated as follows: Based on the precedence rules of this answer choice $'/' > '+' = '*'$, $'/'$ has the highest priority, followed by $'+'$ and $'*'$, which have the same priority. Step1: Since $'/'$ has the highest priority, we calculate $'/'$ first: $9 / 6 = 1.5$, so the expression becomes $1 + 9 * 1.5$. Step2: Since $'+'$ and $'*'$ have the same priority, we evaluate the expression from left to right. $1 + 9 = 10$, so the expression becomes $10 * 1.5$. Step3: Finally, $10 * 1.5 = 15$, so this answer choice evaluates to 15.

For B: The value of $1 + 9 * 9 / 6$ can be calculated as follows: Based on the precedence rules of this answer choice $'*' > '+' = '/'$, $'*'$ has the highest priority, followed by $'+'$ and $'/'$, which have the same priority. Step1: Since $'*'$ has the highest

priority, we calculate '*' first: $9 * 9 = 81$, so the expression becomes $1 + 81 / 6$. Step2: Since '+' and '/' have the same priority, we evaluate the expression from left to right. $1 + 81 = 82$, so the expression becomes $82 / 6$. Step3: Finally, $82 / 6 = 13.666666666666666$, so this answer choice evaluates to 13.666666666666666.

For C: The value of $1 + 9 * 9 / 6$ can be calculated as follows: Based on the precedence rules of this answer choice '+' = '*' > '/', '+' and '*' have the same priority, which is higher than that of '/'. Step1: Since '+' and '*' have the same priority over '/', we evaluate '+' and '*' from left to right. In this case, '+' comes before '*', so we calculate '+' first: $1 + 9 = 10$, so the expression becomes $10 * 9 / 6$. Step2: '*' has priority over '/'. Now we calculate '*': $10 * 9 = 90$, so the expression becomes $90 / 6$. Step3: Finally, $90 / 6 = 15$, so this answer choice evaluates to 15.

For D: The value of $1 + 9 * 9 / 6$ can be calculated as follows: Based on the precedence rules of this answer choice '/' > '+' > '*', '/' has the highest priority, followed by '+', and '*' has the lowest priority. Step1: Since '/' has the highest priority, we calculate '/' first: $9 / 6 = 1.5$, so the expression becomes $1 + 9 * 1.5$. Step2: Since '+' has priority over '*', we then calculate '+': $1 + 9 = 10$, so the expression becomes $10 * 1.5$. Step3: Finally, $10 * 1.5 = 15$, so this answer choice evaluates to 15.

Given the above, only with choice B, the expression evaluates to 13.666666666666666, so the correct answer should be [B].

Appendix C

GPT4 OUTPUTS ON FOUR TYPES OF PROMPTS IN TASK 3

In this appendix, we present the complete responses generated by GPT-4 for an example from Task 3, demonstrating the application of three syntax rules to an English sentence.

C.1 0-shot output

To translate the sentence “The farmer with a bright smile carries the heavy basket carefully across the field.” into Language A, we need to follow the linguistic rules provided.

First, let’s identify the components of the sentence: - Subject: The farmer with a bright smile - Object: the heavy basket - Verb: carries - Adverb modifying the verb: carefully - Prepositional phrase modifying the verb: across the field

According to the rules of Language A, the prepositional phrase modifying the verb is placed before the verb, and if there is an adverb modifying the verb, the sentence follows the word order of prepositional phrase, adverb, and then the verb.

So, the sentence structure in Language A should be: Subject - Object - Prepositional phrase - Adverb - Verb.

Now, let’s translate each word using the provided word pairs and rearrange them according to the rules:

- The farmer = tgt_the tgt_farmer - with a bright smile = tgt_with tgt_a tgt_bright tgt_smile - carries = tgt_carries - the heavy basket = tgt_the tgt_heavy tgt_basket - carefully = tgt_carefully - across the field = tgt_across tgt_the tgt_field

Putting it all together in the correct order for Language A:

[tgt_the tgt_farmer tgt_with tgt_a tgt_bright tgt_smile tgt_the tgt_heavy tgt_basket tgt_across tgt_the tgt_field tgt_carefully tgt_carries].

So the translated sentence in Language A is:

[tgt_the tgt_farmer tgt_with tgt_a tgt_bright tgt_smile tgt_the tgt_heavy tgt_basket
tgt_across tgt_the tgt_field tgt_carefully tgt_carries]. >

C.2 0-shot-CoT output

Let's translate the sentence "The farmer with a bright smile carries the heavy basket carefully across the field" step by step according to the rules of Language A.

1. *Subject-Object-Verb structure:* The canonical word order in Language A is Subject-Object-Verb. In the given sentence, "The farmer" is the subject, "the heavy basket" is the object, and "carries" is the verb. So, we will keep this order in mind for the final structure.

2. *Prepositional phrase modifying a verb:* The prepositional phrase "across the field" modifies the verb "carries". According to the rules, this phrase should be placed before the verb.

3. *Adverb modifying a verb:* The adverb "carefully" modifies the verb "carries". The rule states that if there is a prepositional phrase modifying the verb, the sentence follows the word order of prepositional phrase, adverb, and then the verb. So "carefully" will come after "across the field" and before "carries".

4. *Prepositional phrase modifying a noun:* The prepositional phrase "with a bright smile" modifies the noun "farmer". According to the rules, this phrase should be placed before the noun. If there are determiners or adjectives modifying the noun, the sentence follows the word order of prepositional phrases, determiners, adjectives, and then the noun. Since "the" is a determiner and there are no adjectives modifying "farmer", the order will be "with a bright smile the farmer".

Now, let's put it all together following the rules and translate it into Language A using the provided word pairs:

"The farmer with a bright smile" -> "with a bright smile the farmer" -> "tgt_with
tgt_a tgt_bright tgt_smile tgt_the tgt_farmer" "carries the heavy basket" -> "the heavy
basket carries" -> "tgt_the tgt_heavy tgt_basket tgt_carries" "carefully across the field"
-> "across the field carefully" -> "tgt_across tgt_the tgt_field tgt_carefully"

Combining these segments while maintaining the Subject-Object-Verb order and placing the prepositional phrase and adverb before the verb, we get:

“tgt_with tgt_a tgt_bright tgt_smile tgt_the tgt_farmer tgt_the tgt_heavy tgt_basket tgt_across tgt_the tgt_field tgt_carefully tgt_carries.”

Therefore, the translated sentence in Language A is:

[tgt_with tgt_a tgt_bright tgt_smile tgt_the tgt_farmer tgt_the tgt_heavy tgt_basket tgt_across tgt_the tgt_field tgt_carefully tgt_carries].

C.3 1-shot output

To translate the sentence “The farmer with a bright smile carries the heavy basket carefully across the field.” into Language A, we need to follow the linguistic rules provided:

1. Subject-Object-Verb structure: The subject is “The farmer with a bright smile”, the object is “the heavy basket”, and the verb is “carries”.
2. Prepositional phrases modifying a noun are placed before the noun: “with a bright smile” modifies “The farmer”, so it comes before “The farmer”.
3. Prepositional phrases modifying a verb are placed before the verb: “across the field” modifies “carries”, so it comes before “carries”.
4. Adverbs modifying a verb are placed before the verb: “carefully” modifies “carries”, so it comes before “carries”.
5. The word order for a sentence with both a prepositional phrase and an adverb modifying the verb is prepositional phrase, adverb, and then verb.

Following these rules, the sentence in Language A should be structured as:

“The farmer with a bright smile” -> tgt_with tgt_a tgt_bright tgt_smile tgt_the tgt_farmer
 “the heavy basket” -> tgt_the tgt_heavy tgt_basket
 “carefully across the field carries” -> tgt_carefully tgt_across tgt_the tgt_field tgt_carries

Putting it all together, the translated sentence in Language A is:

[tgt_with tgt_a tgt_bright tgt_smile tgt_the tgt_farmer tgt_the tgt_heavy tgt_basket tgt_carefully tgt_across tgt_the tgt_field tgt_carries .] >

C.4 1-shot-CoT output

To translate the English sentence “The farmer with a bright smile carries the heavy basket carefully across the field.” into Language A, we will follow the linguistic rules provided.

First, we identify the root verb, which is ‘carries’.

Next, we find the subject of the sentence, ‘farmer’, and its related tokens: [‘The’, ‘with a bright smile’].

Then, we locate the object of the sentence, ‘basket’, and its related tokens: [‘the’, ‘heavy’].

We then move the object and its related tokens behind the subject and its related tokens to follow the Subject-Object-Verb word order of Language A: ‘The farmer with a bright smile the heavy basket carries carefully across the field.’.

To apply the ‘Prepositional Phrase before Verb’ transformation, we first identify the verb, which is ‘carries’.

Next, we identify the prepositional phrase that modifies the verb which is ‘across the field’.

We then move the prepositional phrase in front of the verb.

Next, we locate the adverb that modifies the verb, which is ‘carefully’.

Then, we move the adverb ‘carefully’ in front of the verb ‘carries’.

Now, the sentence is in Prepositional Phrase, Adverb, Verb word order as applied in Language A: ‘The farmer with a bright smile the heavy basket across the field carefully carries.’.

Now we translate the sentence to Language A using the word pairs between English and Language A for the example sentence.

Lastly, we have the translated example sentence: [tgt_the tgt_farmer tgt_with tgt_a tgt_bright tgt_smile tgt_the tgt_heavy tgt_basket tgt_across tgt_the tgt_field tgt_carefully tgt_carries]. >