

An In-depth Look at Gemini’s Language Abilities

Syeda Nahida Akter^{*,1}, Zichun Yu^{*,1}, Aashiq Muhamed^{*,1}, Tianyue Ou^{*,1}, Alex Bäuerle¹
 Ángel Alexander Cabrera¹, Krish Dholakia², Chenyan Xiong¹, Graham Neubig¹
¹Carnegie Mellon University, ²BerriAI

Abstract

The recently released Google Gemini class of models are the first to comprehensively report results that rival the OpenAI GPT series across a wide variety of tasks. In this paper, we do an in-depth exploration of Gemini’s language abilities, making two contributions. First, we provide a third-party, objective comparison of the abilities of the OpenAI GPT and Google Gemini models with reproducible code and fully transparent results. Second, we take a closer look at the results, identifying areas where one of the two model classes excels. We perform this analysis over 10 datasets testing a variety of language abilities, including reasoning, answering knowledge-based questions, solving math problems, translating between languages, generating code, and acting as instruction-following agents. From this analysis, we find that Gemini Pro achieves accuracy that is close but slightly inferior to the corresponding GPT 3.5 Turbo on all tasks that we benchmarked. We further provide explanations for some of this under-performance, including failures in mathematical reasoning with many digits, sensitivity to multiple-choice answer ordering, aggressive content filtering, and others. We also identify areas where Gemini demonstrates comparably high performance, including generation into non-English languages, and handling longer and more complex reasoning chains. Code and data for reproduction can be found at <https://github.com/neulab/gemini-benchmark>

1 Introduction

Gemini is the most recent in a series of large language models released by Google DeepMind [Gemini Team, 2023]. It is notable in particular because the results reported by the Gemini team are the first to rival the OpenAI GPT model series [Brown et al., 2020] across a wide variety of tasks. Specifically, Gemini’s “Ultra” version reportedly outperforms GPT-4 on a wide variety of tasks, while Gemini’s “Pro” version is reportedly comparable to GPT-3.5 Gemini Team [2023]. Despite the potential impact of these results, the exact evaluation details and model predictions have not been released, limiting the ability to reproduce, inspect, and analyze the results and their implications in detail.

In this paper, we conduct an in-depth exploration of Gemini’s language understanding and generation abilities, with two goals:

1. We aim to provide a **third-party, objective comparison** of the abilities of the OpenAI GPT and Google Gemini model classes with reproducible code and fully transparent results.
2. We aim to take an **in-depth look into the results**, identifying areas where one of the two model classes excels.

Furthermore, we also perform a limited comparison with the recently released Mixtral model, as a point of reference for a best-in-class open source model [Mistral AI team, 2023].


We perform this analysis over 10 datasets, testing a variety of text understanding and generation capabilities, including the models’ abilities to answer knowledge-based questions (MMLU; Hendrycks

^{*}Lead authors. Individual author contributions are listed in Appendix A.

Task	Model				
	Dataset	Gemini Pro	GPT 3.5 Turbo	GPT 4 Turbo	Mixtral
Knowledge-based QA	MMLU (5-shot)	64.12	<u>67.75</u>	80.48	-
	MMLU (CoT)	60.63	<u>70.07</u>	78.95	-
Reasoning	BIG-Bench-Hard	65.58	<u>71.02</u>	83.90	41.76
Mathematics	GSM8K	69.67	<u>74.60</u>	92.95	58.45
	SVAMP	79.90	<u>82.30</u>	92.50	73.20
	ASDIV	81.53	<u>86.69</u>	91.66	74.95
	MAWPS	95.33	99.17	<u>98.50</u>	89.83
Code Generation	HumanEval	52.44	<u>65.85</u>	73.17	-
	ODEX	38.27	<u>42.60</u>	46.01	-
Machine Translation	FLORES (0-shot)	29.59	<u>37.50</u>	46.57	-
	FLORES (5-shot)	29.00	<u>38.08</u>	48.60	-
Web Agents	WebArena	7.09	<u>8.75</u>	15.16	1.37

Table 1: Main results of our benchmarking. The best model is listed in bold, and the second best model is underlined. Mixtral was only evaluated on a subset of the tasks.

et al. [2021]), perform reasoning (BigBenchHard; Suzgun et al. [2022]), answer mathematics questions (e.g. GSM8K; Cobbe et al. [2021]), translate between languages (e.g. FLORES; Goyal et al. [2022]), generate code (e.g. HumanEval; Chen et al. [2021]), and act as an instruction-following agent (WebArena; Zhou et al. [2023b]).¹

A summary of our main results can be found in Table 1. In sum, we found that across all tasks, as of this writing (December 19, 2023), **Gemini’s Pro model achieved comparable but slightly inferior accuracy compared to the current version of OpenAI’s GPT 3.5 Turbo**. In the following sections, we will detail our experimental methodology (Section 2) and then perform an in-depth description and analysis of the results on each task. Each analysis is accompanied by an online results browser using Zeno [Cabrera et al., 2023],² which can be accessed through the  Zeno Report images in this PDF. All results and code for reproduction can be found at <https://github.com/neulab/gemini-benchmark>.

2 Experimental Setup

Before discussing evaluation results and findings, this section describes our experiment configurations, including models tested, model querying details, and evaluation procedures.

2.1 Models Tested

In this work, we compare 4 models.

Gemini Pro is the second largest model in the Gemini Series, next to the largest Gemini Ultra.³ The model is based on the Transformer [Vaswani et al., 2017] architecture and was trained multimodally over videos, text, and images. The number of parameters and size of training data are not disclosed. In the original Google paper on Gemini, it was reported to achieve similar performance to GPT 3.5 Turbo.

GPT 3.5 Turbo is the second most capable text model served by OpenAI, part of the GPT-3 series [Brown et al., 2020]. The model has been instruction tuned and trained using reinforcement learning

¹Note that Gemini is a multi-modal model, but for this examination, we only focus on Gemini’s language understanding, generation, and translation abilities.

²<https://zenoml.com>

³Gemini Ultra is not yet publicly available, and thus we do not test it in the current version of this paper.

from human feedback [Ouyang et al., 2022], but was trained solely on text. Similarly, model size and precise training details are not disclosed.

GPT 4 Turbo is the second generation of the GPT-4 [OpenAI, 2023] family, a family of models trained multimodally. The turbo version is moderately cheaper than the original GPT-4 model (making it more conducive to benchmarking) and similarly lacks detail of the actual training algorithms, data, or parameter size.

Mixtral in contrast, is an open-source mixture-of-experts model, consisting of eight 7B parameter models [Mistral AI team, 2023]. It has been reported to achieve comparable accuracy to GPT 3.5 Turbo on several tasks, including some examined in this paper.

2.2 Model Querying Details

All models were queried through the unified interface provided by LiteLLM⁴ between December 11-15, 2023. Gemini was queried through Google Vertex AI, OpenAI models through the OpenAI API, and Mixtral through the API provided by Together.⁵ For reference, we also list the current pricing of each model through these APIs for 1M tokens in Table 2, which provides an approximate measure of how efficiently the models can be run.

It is also notable that in some cases Gemini Pro blocks some questions, particularly in the case of potentially illegal or sensitive material. Responses were blocked for some portion of the testing examples, so we treated these examples as incorrect. For each task where a significant number of responses were blocked, we will discuss this effect in the corresponding section.

Language Model	Input	Output
Gemini Pro	\$1.00	\$2.00
GPT-3.5 Turbo	\$1.00	\$2.00
GPT-4	\$10.00	\$30.00
Mixtral	\$0.60	\$0.60

Table 2: Pricing per 1M tokens. Gemini Pro charges by character; we multiply by 4, a rule-of-thumb average of characters per English token [Raf, 2023].

2.3 Evaluation Procedure

To perform a fair comparison between the models, we re-ran experiments with all models using *exactly the same prompts and evaluation protocol for all evaluated models*. We make this decision to ensure that all models are compared on exactly the same footing, in contrast to previous papers where these settings may differ. In general, we tried to follow both prompts and evaluators from standard repositories, either those officially released by the datasets themselves, or from the Eleuther evaluation harness [Gao et al., 2023a]. These prompts generally consist of a query, input, and few-shot examples, sometimes including chain-of-thought reasoning [Wei et al., 2022]. In some cases, we found it necessary to make small changes from standard practice to stably evaluate all models under consideration; all such deviations are noted below and implemented in the companion code repository.

3 Knowledge-based QA Zeno Report

In this category, we focus on 57 knowledge-based multiple-choice question-answering tasks from MMLU [Hendrycks et al., 2021], which span topics across STEM, the humanities, the social sciences, and more. MMLU has been widely used as a holistic evaluation of LLMs’ knowledge-based capabilities. There are 14,042 test samples in total.

3.1 Experimental Details

Generation Parameters We examine two popular evaluation methods in this task, including the standard 5-shot prompts from Hendrycks et al. [2021] and 5-shot chain-of-thought prompts from chain-of-thought-hub⁶ [Fu et al., 2023] with a prefix of “Let’s think step by step.” [Kojima et al.,

⁴<https://litellm.ai/>

⁵<https://cloud.google.com/vertex-ai/docs> <https://openai.com/api> <https://docs.together.ai/docs>

⁶<https://github.com/FranxYao/chain-of-thought-hub>

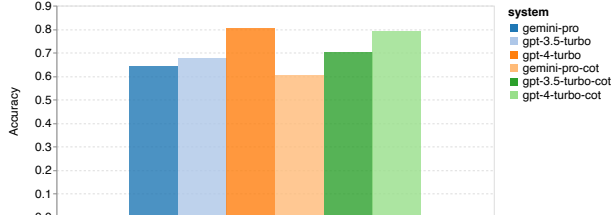


Figure 1: Overall accuracy on MMLU with 5-shot prompts and chain-of-thought prompts

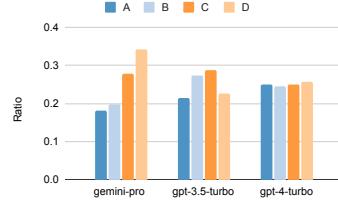


Figure 2: Ratio of multiple-choice answers being predicted by models

2022]. Note that we opt not to sample multiple responses and perform self-consistency based reranking [Wang et al., 2022a] as done by Gemini Team [2023], as this significantly increases cost and may not be feasible in many scenarios. We generate via greedy search with a temperature of 0.

Evaluation For the standard prompting, we directly take the first character generated by models as their answer since this is what the 5-shot prompts imply. Sometimes, the model may not follow this format and output the answer elsewhere. We treat examples like this as incorrect (and elaborate more on the effect of this in the following section). For the chain-of-thought prompting, we perform answer extraction from the model’s response and set the default answer as “C” if no answer can be extracted, as is done in chain-of-thought-hub.

3.2 Results and Analysis

In this section, we compare and analyze the overall performance, performance by sub-tasks, and performance by output length on MMLU.

First, from the overall results shown in Figure 1, we can see that Gemini Pro achieves an accuracy lower than that of GPT 3.5 Turbo, and much lower than that of GPT 4 Turbo. We saw little difference in performance using chain-of-thought prompting, likely because MMLU is mostly a knowledge-based question answering task that may not benefit significantly from stronger reasoning-oriented prompts.⁷

Based on this overall result, we next dive a bit deeper. One first notable point is that all questions in MMLU are multiple-choice with 4 potential answers ordered A through D. In Figure 2, we show the ratio of the number of times each model selects each multiple choice answer. From this figure, we can see that Gemini has a very skewed label distribution, biased towards selecting the final choice of “D”, which contrasts to the result of the GPT model, which is more balanced. This may indicate that Gemini has not been heavily instruction-tuned towards solving multiple-choice questions, which can cause models to be biased with respect to answer ordering [Tjauatja et al., 2023].

Next, we examine each subtask’s performance. Figure 3 illustrates each model’s performance on selected representative tasks. We notice that Gemini Pro underperforms on most tasks compared to GPT 3.5. Chain-of-thought prompting decreases the variance across the subtasks.

Further, we dig deeper into the tasks where Gemini Pro underperforms/outperforms GPT 3.5 the most. From Figure 4, we can observe that Gemini Pro falls behind GPT 3.5 on `human_sexuality` (social sciences), `formal_logic` (Humanities), `elementary_mathematics` (STEM), and `professional_medicine` (specialized domains). For the two tasks where Gemini Pro outperformed GPT 3.5 Turbo, gains were marginal.

The underperformance of Gemini Pro on particular tasks can be attributed to two reasons. First, as previously mentioned in subsection 2.2, in some cases Gemini fails to return an answer. In most MMLU sub-tasks, the API response rate was greater than 95%, but two had notably low response rates: `moral_scenarios` at 85% and `human_sexuality` at 28%. This indicates that low performance on some tasks can be attributed to content filters on the input. Second, Gemini Pro performed somewhat more poorly at the basic mathematical reasoning necessary to solve the `formal_logic` and `elementary_mathematics` tasks, which we examine further in Section 4.

⁷Note that our evaluation numbers for GPT 4 Turbo (80.5%) are slightly worse than those from GPT 4 reported by OpenAI [2023] (86.4%). This drop could likely be attributed to the “Turbo” model, but is also possibly the result of slight differences in prompting or generation methods.

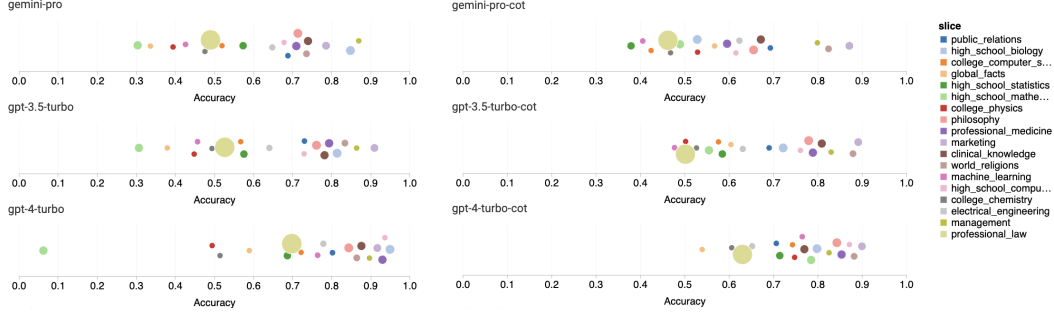
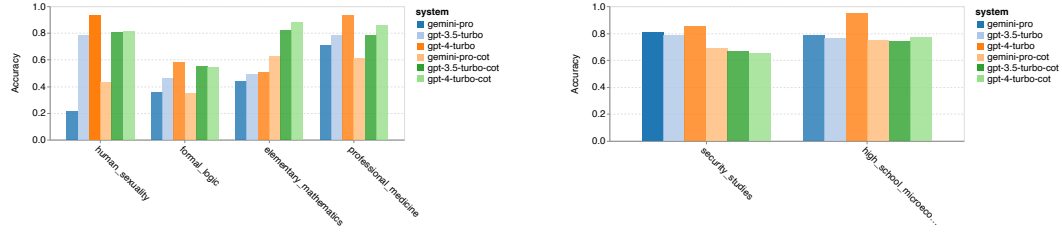


Figure 3: Accuracy by each subtask on MMLU

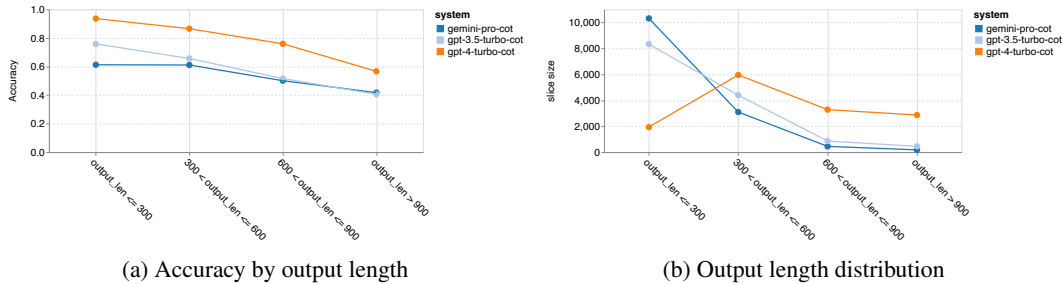


(a) Top-4 tasks where GPT 3.5 wins over Gemini Pro

(b) Tasks where Gemini Pro wins over GPT 3.5

Figure 4: Tasks where Gemini Pro and GPT 3.5 prevail on MMLU

Finally, we analyze how the output length in the chain-of-thought prompting affects the model performance in Figure 5. Generally, a stronger model tends to perform more complex reasoning and thus outputs a longer response. One of the noteworthy advantages of Gemini Pro is that its accuracy is less influenced by the output length compared to the two counterparts. It even outperforms GPT 3.5 when the output length is over 900. However, it also can be seen that Gemini Pro and GPT 3.5 Turbo rarely output these long reasoning chains compared to GPT 4 Turbo.



(a) Accuracy by output length

(b) Output length distribution

Figure 5: Analysis of output length on MMLU

4 General-purpose Reasoning

In this category, we focus on 27 diverse reasoning tasks from BIG-Bench Hard [Suzgun et al., 2022] which consists of arithmetic, symbolic and multilingual reasoning and factual knowledge understanding tasks. Most of the tasks consist of 250 question-answer pairs, with a few having somewhat fewer.

4.1 Experimental Details

Generation Parameters We follow standard 3-shot prompts from the Eleuther harness across all models where each question is followed by a chain of thought resulting in a final concluding sentence

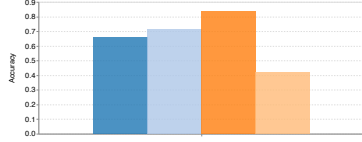


Figure 6: Overall accuracy on BIG-Bench-Hard

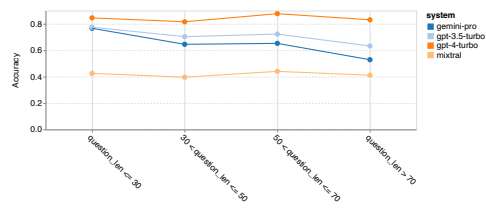


Figure 7: Accuracy by question length on BIG-Bench-Hard

of “So the answer is ____.”. For hyperparameters, we perform greedy decoding, generating with temperature of 0.

Evaluation The Eleuther evaluation harness implementation of BIG-Bench Hard matches the sentence “So the answer is ____.” and extracts the text. However, we found that for some models, they did not produce this sentence verbatim, even in cases when they generated the correct answer, particularly multiple-choice tasks where the answer is an option chosen from the question text (e.g., “answer: (B)”). To remedy this, we modified the matching rule, instead taking the last word of the generated text as the answer of the question only for multiple-choice tasks.

4.2 Results and Analysis

For the reasoning tasks, we report the overall performance, performance by question complexity, and performance by BIG-Bench sub-task.

First, we illustrate the overall accuracy in Figure 6, we can see that Gemini Pro achieves an accuracy slightly lower than that of GPT 3.5 Turbo, and much lower than that of GPT 4 Turbo. In contrast, the Mixtral model achieves much lower accuracy.

Based on this overall result, let us dig a little bit deeper into why Gemini might be underperforming. First, we examined *accuracy by the length of the question*, as detailed in Figure 7. We found that Gemini Pro underperformed on longer, more complex questions while the GPT models were more robust to this. This was particularly the case for GPT 4 Turbo, which showed very little degradation even on longer questions, indicating an impressively robust ability to understand longer and more complex queries. GPT 3.5 Turbo fell in the middle with respect to this robustness. Mixtral was notably stable with respect to question length, but had low accuracy overall.

Next we look at whether there are variations in accuracy by the specific task in BIG-Bench-Hard. Below, we list the tasks where GPT 3.5 Turbo outperformed Gemini Pro by the largest amount.

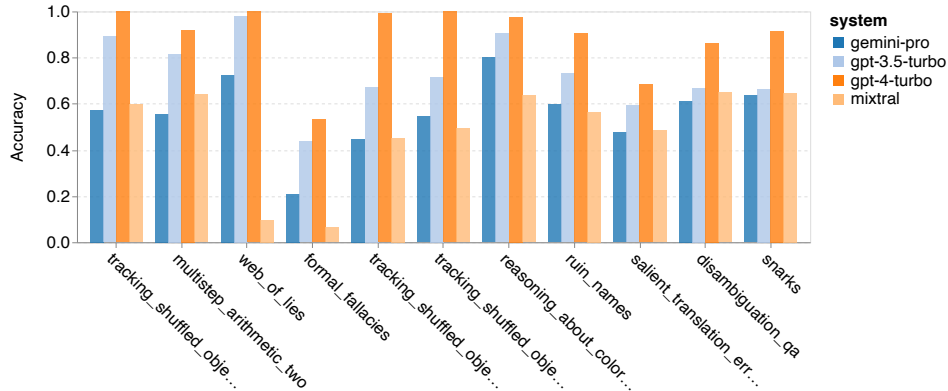


Figure 8: Tasks where GPT 3.5 Turbo excels over Gemini Pro

We can notice that Gemini Pro is particularly bad at the ‘tracking_shuffled_objects’ tasks. These tasks involve keeping track of who has certain objects as they are traded among people, and Gemini Pro often has difficulty keeping the order straight (as Figure 8).

Prompt	<p>Alice, Bob, and Claire are friends and avid readers who occasionally trade books. At the start of the semester, they each buy one new book: Alice gets Ulysses, Bob gets Frankenstein, and Claire gets Lolita.</p> <p>As the semester proceeds, they start trading around the new books. First, Claire and Bob swap books. Then, Bob and Alice swap books. Finally, Claire and Bob swap books. At the end of the semester, Bob has</p> <p>Options:</p> <p>(A) Ulysses</p> <p>(B) Frankenstein</p> <p>(C) Lolita</p>
Label	(B)
Output	<p>(0) At the start: Alice: Ulysses, Bob: Frankenstein, Claire: Lolita.</p> <p>(1) Claire and Bob swap books: Alice: Ulysses, Bob: Lolita, Claire: Frankenstein.</p> <p>(2) Bob and Alice swap books: Alice: Frankenstein, Bob: Ulysses, Claire: Lolita.</p> <p>(3) Claire and Bob swap books: Alice: Frankenstein, Bob: Lolita, Claire: Ulysses.</p> <p>At the end of the semester, Bob has Lolita. So the answer is (C).</p>

Table 3: Example of Gemini Pro’s inability to maintain state in `tracking_shuffled_objects`.

In some tasks, namely `multistep_arithmetic_two`, `salient_translation_error_detection`, `snarks`, `disambiguation_qa` and two of `tracking_shuffled_objects` tasks, Gemini Pro even performs worse than the Mixtral model.

However, there were a few tasks where Gemini Pro outperformed GPT 3.5 Turbo. The Figure 9 shows the six tasks where Gemini Pro outperformed GPT 3.5 Turbo by the largest amount. These were heterogeneous and included those that required world knowledge (`sports_understanding`), manipulating stacks of symbols (`dyck_languages`), sorting words in alphabetical order (`word_sorting`), and parsing tables (`penguins_in_a_table`), among others.

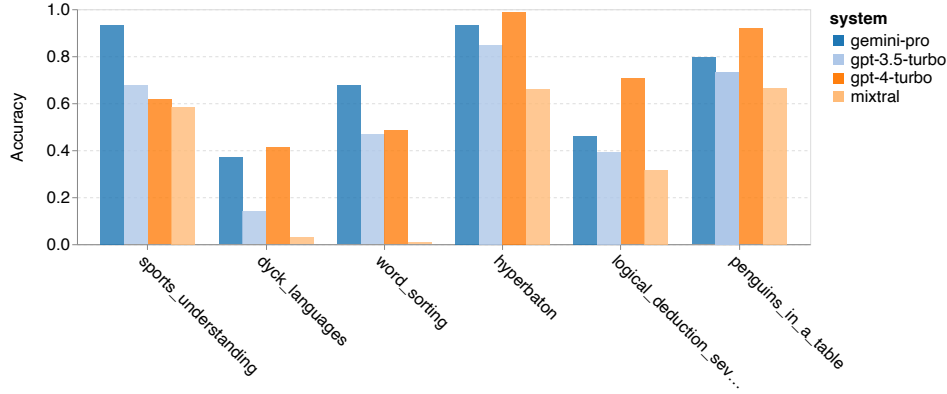


Figure 9: Tasks where Gemini Pro excels over GPT 3.5 Turbo

We further investigate the robustness of LLMs across different answer types in the Figure below. We can see that Gemini Pro shows the worst performance in Valid/Invalid answer type which falls under the task `formal_fallacies`. Interestingly **68.4%** of questions from this task were blocked by. However, Gemini outperformed all GPT models as well as Mixtral by a significant margin on Other answer types (consisting of the `word_sorting` and `dyck_language` tasks) which follows a similar line of findings as above i.e., Gemini is particularly good at word rearrangement and producing symbols in the correct order. Also for MCQ answers, **4.39%** questions were blocked Gemini Pro, and while GPT models excel in this genre, Gemini struggles to compete with them.

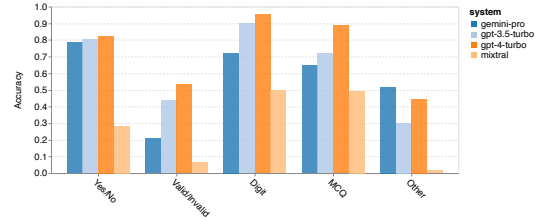


Figure 10: Accuracy by answer types

In sum, there did not seem to be a particularly strong trend in which tasks one model performed better than the other, so when performing general-purpose reasoning tasks it may be worth trying both the Gemini and GPT models before making a decision on which to use.

5 Mathematics Zeno Report

To evaluate the mathematical reasoning ability of the evaluated models, we explore four math word problems benchmarks (1) the grade-school math benchmark, GSM8K [Cobbe et al., 2021], (2) the SVAMP dataset [Patel et al., 2021] with questions generated by varying word-order to check the robust reasoning ability, (3) the ASDIV dataset [Miao et al., 2020] with diverse language patterns and problem types and (4) the MAWPS benchmark [Koncel-Kedziorski et al., 2016] consisting of arithmetic and algebraic word problems.

5.1 Experimental Details

Generation Parameters We consider standard 8-shot chain-of-thought prompts [Gao et al., 2023a, Wei et al., 2022] where each question in few-shot prompting is associated with a chain of thought for generating the corresponding answer. We evaluate all LLMs via greedy decoding using a temperature of 0.

Evaluation In evaluation, we make a slight modification to the standard evaluation protocol in the Eleuther harness, which consisted of matching the words “The answer is” followed by a numerical output. We found that all evaluated models had a tendency to output the correct answer even when this specific phrase was not present. To mitigate this, we simply taking the last number of the generated text as the answer of the question, which resulted in higher accuracy overall.

5.2 Results and Analysis

In this section, we compare the accuracy of Gemini Pro to GPT 3.5 Turbo, GPT 4 Turbo, and Mixtral, on the four math word problems tasks, examining overall performance, performance by question complexity, and performance by chain-of-thought depth.

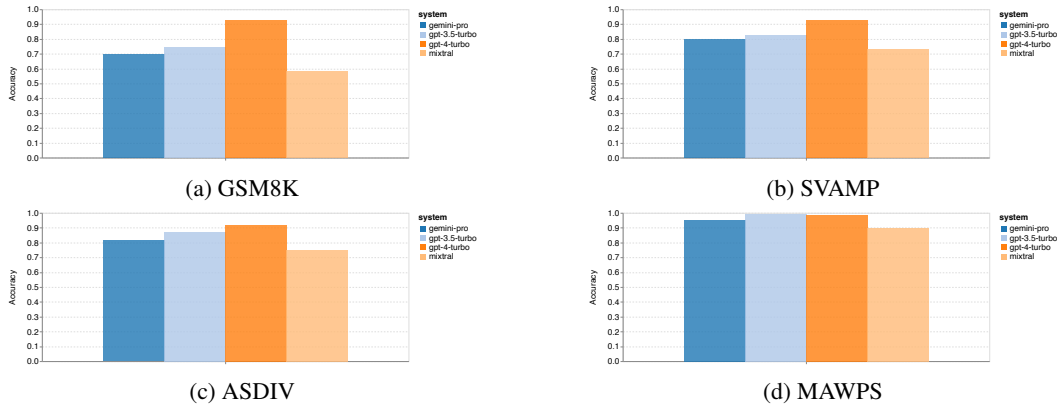


Figure 11: Overall accuracy across four mathematical reasoning tasks

First, looking at overall results in the Figure 11, we can see that Gemini Pro achieves an accuracy slightly lower than that of GPT 3.5 Turbo, and much lower than that of GPT 4 Turbo on the GSM8K, SVAMP and ASDIV tasks, which all contain diverse language patterns. For the MAWPS task, all models achieve more than 90% accuracy, although Gemini Pro is still slightly worse than GPT models. Interestingly in this task GPT 3.5 Turbo outperforms GPT 4 Turbo by a close margin. In contrast, the Mixtral model achieves much lower accuracy compared to others.

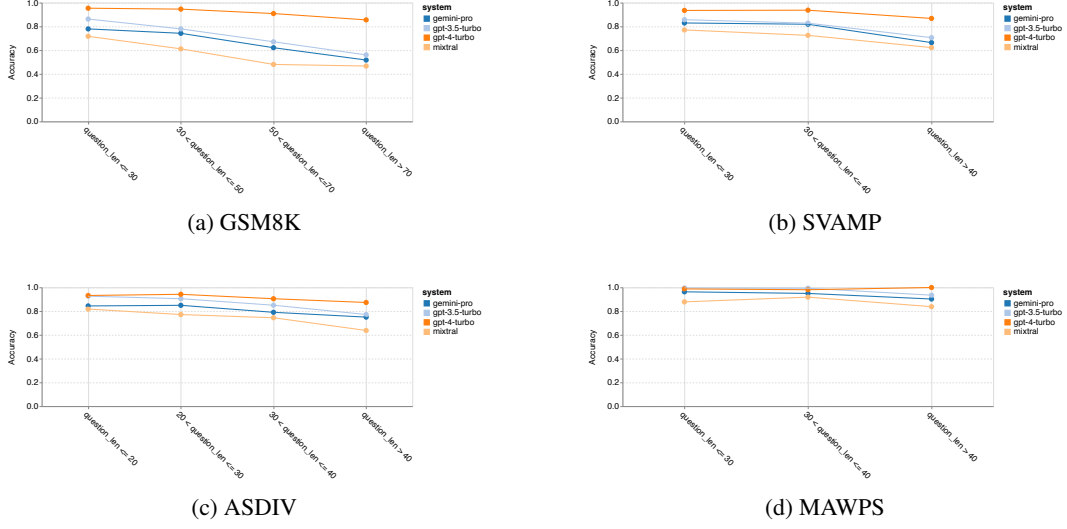


Figure 12: Accuracy by question length across four mathematical reasoning tasks

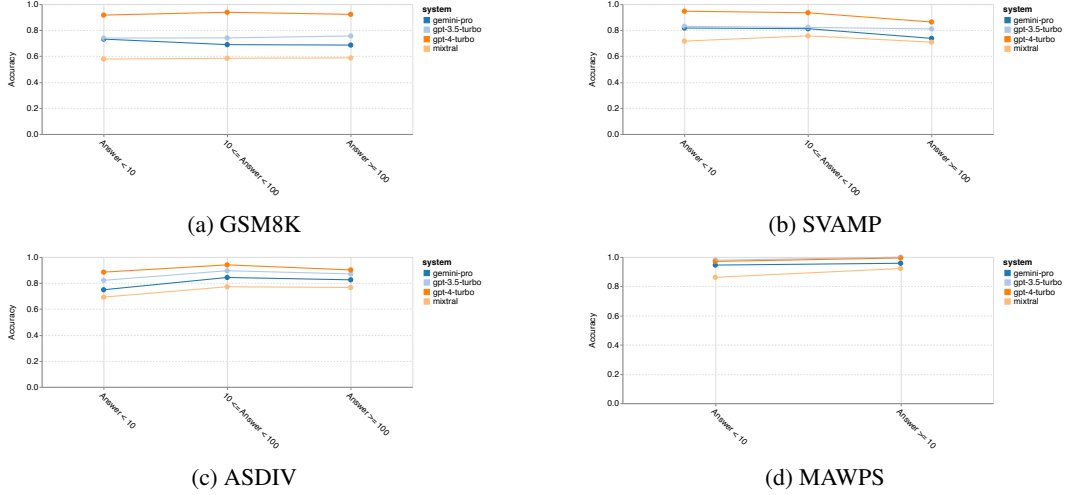


Figure 14: Accuracy by number of digits in the answer across four mathematical reasoning tasks

Similarly to Section 4 we break down the results to observe the robustness of each model to question length in Figure 12. As with the reasoning tasks on BIG-Bench Hard, we see a drop-off on longer questions. As before, GPT 3.5 Turbo outperforms Gemini Pro on shorter questions, but drops off more quickly, with Gemini Pro achieving similar (but still slightly inferior) accuracy on longer questions.

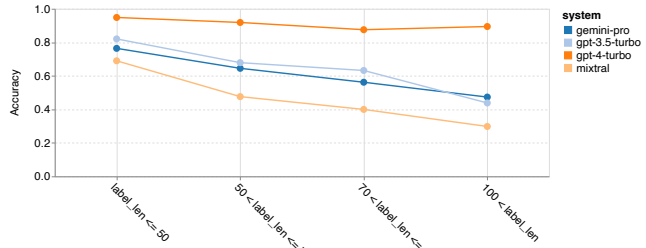


Figure 13: GSM8K accuracy by chain-of-thought length

Additionally, we observe the accuracy of the LLMs when the answer requires longer chains of thought. As shown in Figure 13, GPT 4 Turbo is very robust even when using long reasoning chains, where GPT 3.5 Turbo, Gemini Pro and Mixtral struggle with increasing COT lengths. In this analysis we also find that Gemini Pro is superior to GPT 3.5 Turbo in the most complex examples where the COT length is over 100, but underperforms in the shorter examples.

Finally, we investigate the accuracy of the compared models in generating answers with varying numbers of digits. We create three buckets based on the number of digits in the answer, 1, 2, or 3+ (except for the MAWPS task which does not have answers more than two digits). As shown in [Figure 14](#), GPT 3.5 Turbo appears to be more robust to multi-digit math problems, where Gemini Pro degrades somewhat more on problems with more digits.

6 Code Generation Zeno Report

In this category, we examine the models’ coding abilities using two code generation datasets HumanEval [Chen et al., 2021] and ODEX [Wang et al., 2022b]. The former tests basic code understanding on a limited set of functions from the Python standard library, while the latter tests the ability to use a broader set of libraries from the entire Python ecosystem. Both of them take as input a human-written task description in English (often with test cases). These problems evaluate comprehension of language, algorithmic understanding, and elementary mathematics. Overall, HumanEval has 164 test samples, and ODEX has 439 test samples.

6.1 Experimental Details

Generation Parameters We follow the standard zero-shot code evaluation pipeline provided by the ODEX⁸. We take its recommended hyperparameters with a temperature of 0.8 and top_p of 0.95. We also add a customized instruction “Complete the given code with no more explanation. Remember that there is a 4-space indent before the first line of your generated code.” to ensure that the models’ output fits the desired format.

Evaluation We perform evaluation-based execution, measuring the Pass@1 metric, which determines whether a single sample from the model passes test cases [Chen et al., 2021]. Since code generation is evaluated in a zero-shot fashion, the model may inevitably output code that does not conform to our input format well. Therefore, we perform rudimentary post-processing to regulate the output code and make it fit into the final verification pipeline as much as possible, including the removal of markdown code blocks, the extraction of function implementations and the truncation of stop tokens. It’s noteworthy that for incorrectly formatted indent, we do not manage to manually fix such issues and instead regard them as typical syntax errors.

6.2 Results and Analysis

In this section, we examine the overall performance and present a case study on the code generation.

First, from the overall results shown in [Figure 15](#), we can see that Gemini Pro achieves a Pass@1 lower than GPT 3.5 Turbo and much lower than GPT 4 Turbo on both tasks. The results demonstrate that Gemini’s code generation capabilities still have room for improvement.

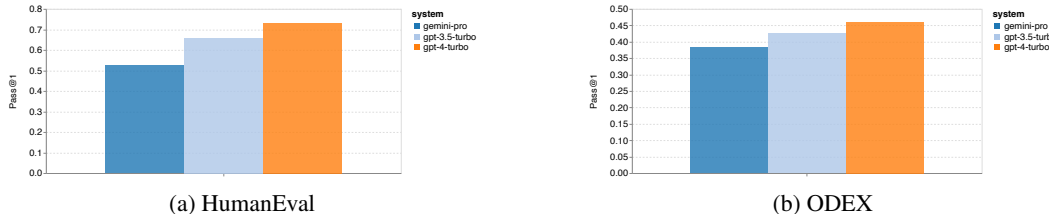
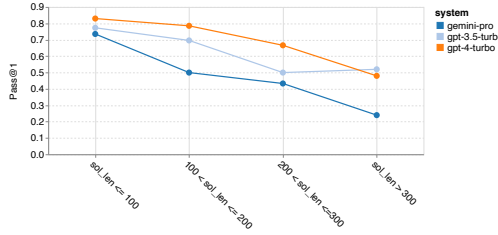


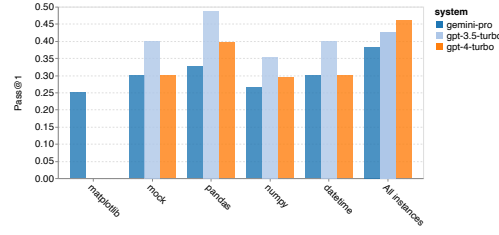
Figure 15: Overall accuracy on code generation tasks

Second, we analyze the relationship between the gold solution length and the model performance in [Figure 16a](#). The solution length can partly indicate the difficulty of solving the corresponding code generation task. We find that even though Gemini Pro achieves comparable Pass@1 with GPT 3.5 when the solution length is below 100 (e.g., easier cases), it falls behind by large margins when the solution becomes longer. This is an interesting contrast to the results from previous sections, where

⁸<https://github.com/zorazrw/odex>



(a) Accuracy by gold solution length on HumanEval



(b) Accuracy by used libraries on ODEX

Figure 16: Comparison of Pass@1 w.r.t. gold solution length and the libraries used by gold solution

we found that in general Gemini Pro performed robustly with respect to longer inputs and outputs on English language tasks.

We also present the analysis of how the libraries required in each solution affect the model performance in Figure 16b. Gemini Pro performs worse than GPT 3.5 on most library-used cases, such as mock, pandas, numpy, and datetime. However, it outperforms GPT 3.5 and GPT 4 on the matplotlib cases, showing stronger capabilities when performing drawing visualization via code.

Finally, we show several concrete examples of failure cases where Gemini Pro performs worse in code generation than GPT 3.5. First, we noticed that Gemini is somewhat worse at correctly choosing functions and arguments from the Python API. For instance, given this prompt:

```
def f_3283984():
    """decode a hex string '4a4b4c' to UTF-8."""
```

Gemini Pro generated the following code, which results in a type mismatch error:

```
bytes(bytearray.fromhex('4a4b4c'), 'utf-8')
```

In contrast, GPT 3.5 Turbo used the following code, which achieves the desired result:

```
hex_string = '4a4b4c'
decoded_string = bytes.fromhex(hex_string).decode('utf-8')
return decoded_string
```

Further, Gemini Pro had a higher proportion of mistakes where the implemented code was syntactically correct but did not correctly match with a more complex intent. For instance, with respect to the following prompt:

```
from typing import List

def remove_duplicates(numbers: List[int]) -> List[int]:
    """From a list of integers, remove all elements that occur more than once.
    Keep order of elements left the same as in the input.
    >>> remove_duplicates([1, 2, 3, 2, 4])
    [1, 3, 4]
    """
```

Gemini Pro created an implementation that just extracts the unique numbers without removing those that appear more than once.

```
seen_numbers = set()
unique_numbers = []
for number in numbers:
    if number not in seen_numbers:
        unique_numbers.append(number)
        seen_numbers.add(number)
return unique_numbers
```

Lang.	Gemini Pro	GPT 3.5 Turbo	GPT 4 Turbo	Google	NLLB
ssw_Latin	5.71	11.15	<u>37.20</u>	-	43.30
sna_Latin	4.36	16.07	42.82	44.40	<u>43.40</u>
ckb_Arab	0.01	19.42	39.58	47.70	<u>47.20</u>
mag_Deva	<u>40.37</u>	38.91	23.83	-	58.50
ibo_Latin	<u>5.26</u>	14.05	41.57	43.50	<u>41.40</u>
hau_Latin	37.32	23.62	49.92	<u>53.20</u>	53.50
pbt_Arab	0.33	20.32	<u>33.22</u>	-	39.40
tam_Tamil	0.01	34.52	48.87	55.80	<u>53.70</u>
kat_Geor	0.04	34.21	43.21	51.40	<u>48.10</u>
gle_Latin	4.60	47.66	56.81	60.10	<u>58.00</u>
kmr_Latin	10.78	25.98	31.96	40.00	<u>39.30</u>
war_Latin	59.14	49.94	54.92	-	<u>57.40</u>
ajp_Arab	<u>49.79</u>	46.93	46.45	-	51.30
lim_Latin	39.90	40.30	<u>41.32</u>	-	47.90
ukr_Cryl	<u>57.92</u>	54.67	57.09	58.60	56.30
fra_Latin	<u>71.46</u>	71.15	70.92	72.70	69.70
lvs_Latin	59.67	55.01	<u>58.05</u>	-	54.80
ron_Latin	65.58	64.20	64.58	<u>65.00</u>	61.30
tpi_Latin	31.13	37.33	48.36	-	<u>41.60</u>
acm_Arab	48.47	<u>44.50</u>	40.71	-	31.90

Table 4: Machine translation performance (chRF (%) scores) across models for all languages using 0-shot prompt. Best scores are bolded, second best underlined.

7 Machine Translation

This set of experiments evaluates the models’ multilingual ability, specifically their ability to translate between various language pairs, using the FLORES-200 machine translation benchmark [NLLB Team et al., 2022]. We focus on a diverse subset of 20 languages used by the analysis of Robinson et al. [2023], which encompass various levels of resource availability and translation difficulty. We evaluate on the 1012 sentences from the test set for all the chosen language pairs. As the first step of this study, we limited our scope to translations from English to other languages (ENG→X) only.

7.1 Experimental Details

Generation Parameters We investigate the efficacy of zero-shot and five-shot prompting strategies across 20 language pairs. Following the guidelines proposed by Gao et al. [2023b], we utilized designated prompts for both zero-shot and few-shot machine translation (MT), as exemplified in Table 8. These prompt settings have shown distinct advantages for users of large language models (LLMs), as noted by Robinson et al. [2023]. Our experimental setup employed a top_p value of 1, a temperature of 0.3, a context_length of -1, and max_tokens 500 to optimize the performance of the translation model.

Evaluation To evaluate the outputs, we utilized the following metrics:

spBLEU: BLEU, a standard in machine translation evaluation, was employed [Papineni et al., 2002]. We computed spBLEU scores following the methodology outlined in Goyal et al. [2022], using the sacreBLEU toolkit [Post, 2018] with the SPM-200 tokenizer [NLLB Team et al., 2022].

chrF2++: Our primary metric, chrF2++, leverages the implementation provided by sacreBLEU [Post, 2018]. This choice is motivated by its ability to address some of BLEU’s limitations. For simplicity, we refer to this metric as chrF in our discussion [Popović, 2017].

Lang.	Gemini Pro	GPT 3.5 Turbo	GPT 4 Turbo	Google	NLLB
ssw_Latin	19.74	7.62	<u>38.07</u>	-	43.30
sna_Latin	4.36	15.84	42.95	44.40	<u>43.40</u>
ckb_Arab	0.01	24.56	40.71	47.70	<u>47.20</u>
mag_Deva	<u>47.54</u>	39.25	45.33	-	58.50
ibo_Latin	<u>5.26</u>	16.29	<u>41.65</u>	43.50	41.40
hau_Latin	5.32	24.22	50.11	<u>53.20</u>	53.50
pbt_Arab	0.33	21.35	<u>34.11</u>	-	39.40
tam_Tamil	0.01	34.86	48.69	55.80	<u>53.70</u>
kat_Geor	0.04	33.61	43.17	51.40	<u>48.10</u>
gle_Latin	4.60	47.30	57.25	60.10	<u>58.00</u>
kmr_Latin	4.91	26.10	32.76	40.00	<u>39.30</u>
war_Latin	48.94	50.85	<u>56.26</u>	-	57.40
ajp_Arab	<u>50.72</u>	47.49	48.12	-	51.30
lim_Latin	<u>46.92</u>	43.25	45.21	-	47.90
ukr_Cryl	<u>57.79</u>	55.19	56.85	58.60	56.30
fra_Latin	<u>71.80</u>	71.34	70.79	72.70	69.70
lvs_Latin	59.93	55.05	<u>58.34</u>	-	54.80
ron_Latin	66.11	64.19	64.42	<u>65.00</u>	61.30
tpi_Latin	35.90	37.39	50.67	-	41.60
acm_Arab	49.78	45.88	<u>46.45</u>	-	31.90

Table 5: Machine translation performance (chRF (%) scores) models for all languages using 5-shot prompt. Best scores are bolded, second best underlined.

7.2 Results and Analysis

Overall Performances In Table 4 and Table 5, we conduct a comparative analysis of Gemini Pro, GPT 3.5 Turbo, and GPT 4 Turbo against established systems like Google Translate.⁹ Additionally, we benchmark against NLLB-MoE [NLLB Team et al., 2022], the leading open-source machine translation (MT) model known for its extensive language coverage. The results indicate that Google Translate generally outperforms other models, and excels in 9 languages, followed by NLLB which excels on 6 and 8 language in 0/5-shot settings. The general language models showed competitive performances but have not yet surpassed the dedicated machine translation systems in translation into non-English languages.

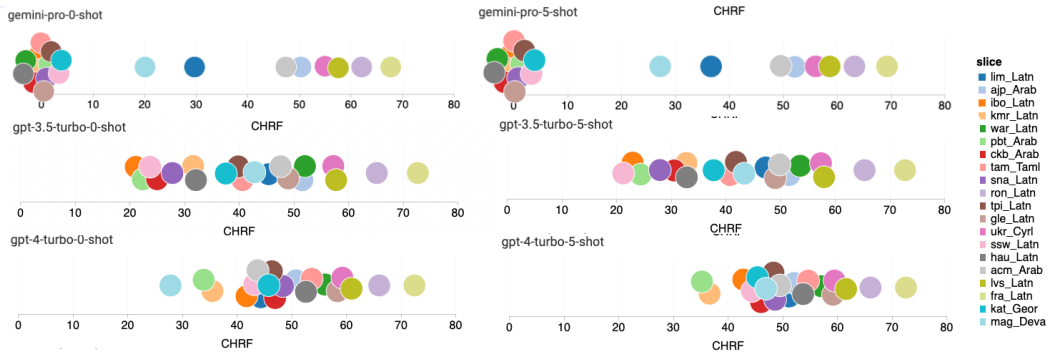


Figure 17: Machine translation performance (chRF (%) scores) by language pairs

Figure 17 illustrates the comparative performance of general language models across language pairs. GPT 4 Turbo showed a consistent deviation of performance with NLLB relative to GPT 3.5 Turbo and Gemini Pro. This reflects the findings in the literature on GPT 4 Turbo’s multilingual performance [OpenAI [2023]]. GPT 4 Turbo also offered larger improvements for low-resource languages (as measured in [NLLB Team et al. [2022]]), whereas for high-resource languages performance was

⁹<http://translate.google.com>

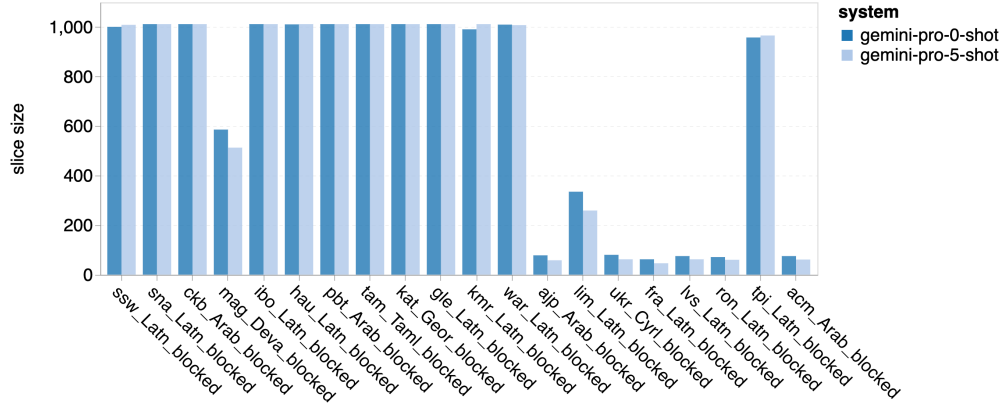


Figure 18: Number of samples that are blocked by Gemini Pro

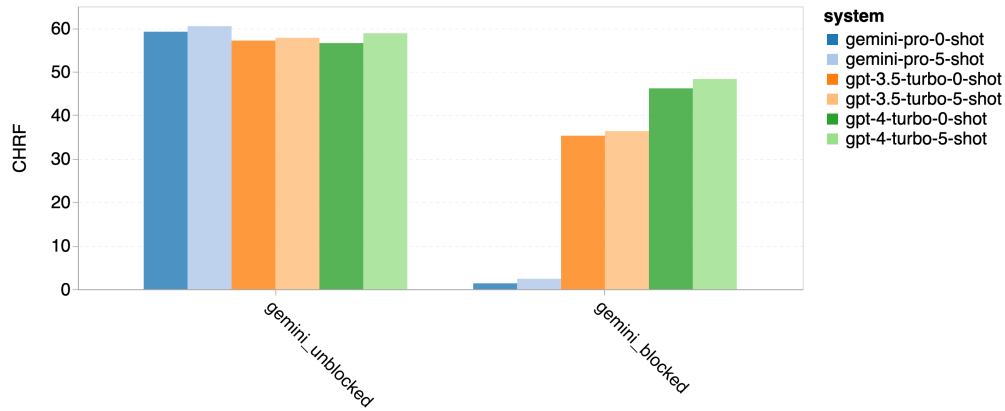


Figure 19: Performance in chrF (%) on blocked and unblocked samples

similar between the LLMs. In comparison, Gemini Pro outperforms both GPT 3.5 Turbo and GPT 4 Turbo on 8 out of 20 languages, and achieved the top performances on 4 languages. However, Gemini Pro showed a strong tendency to block responses in approximately 10 language pairs, which we further study in the next analysis.

Gemini blocked responses Figure 18 highlights that Gemini Pro’s lower performance in these languages is due to its tendency to block responses in scenarios of lower confidence. A response is deemed "blocked" if Gemini Pro in either its 0-shot or 5-shot configuration generates a *Blocked Response* error. A closer examination in Figure 19 reveals that Gemini Pro marginally outperforms GPT 3.5 Turbo and GPT 4 Turbo in unblocked samples where it demonstrates higher confidence. Specifically, it surpasses GPT 4 Turbo by 1.6 chrF in 5-shot and 2.6 chrF in 0-shot settings, and exceeds GPT 3.5 Turbo by 2.7 chrF and 2 chrF in 5-shot and 0-shot settings, respectively. However, our initial analysis of GPT 4 Turbo and GPT 3.5 Turbo’s performance on these samples indicates they are typically more challenging to translate. Gemini Pro’s subpar

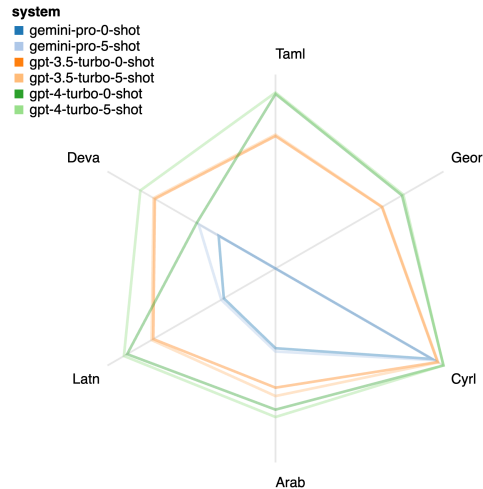


Figure 20: Performance (chrF (%)) by script

performance on these particular samples is especially apparent in instances where the Gemini Pro 0-shot blocks responses but the 5-shot does not, and vice versa.

Other trends Throughout our analysis of the models, we observed that few-shot prompts generally yield a modest enhancement in average performance, with an increasing variance pattern following the order: GPT 4 Turbo < GPT 3.5 Turbo < Gemini Pro. While Gemini Pro’s 5-shot prompts show improvement over its 0-shot counterparts in languages where it demonstrates confidence, in certain languages, such as hau_Latin, the model exhibits significantly reduced confidence, resulting in blocked responses (refer to Table 5).

In Figure 20, we present apparent trends when categorizing languages by family or script. A key observation is Gemini Pro’s competitive performance with other models on Cyrillic scripts, is contrasted by its underperformance on other scripts. GPT-4 stands out, outperforming other models across various scripts, with few-shot prompts being particularly effective. This effectiveness is especially pronounced in languages using the Devanagari script.

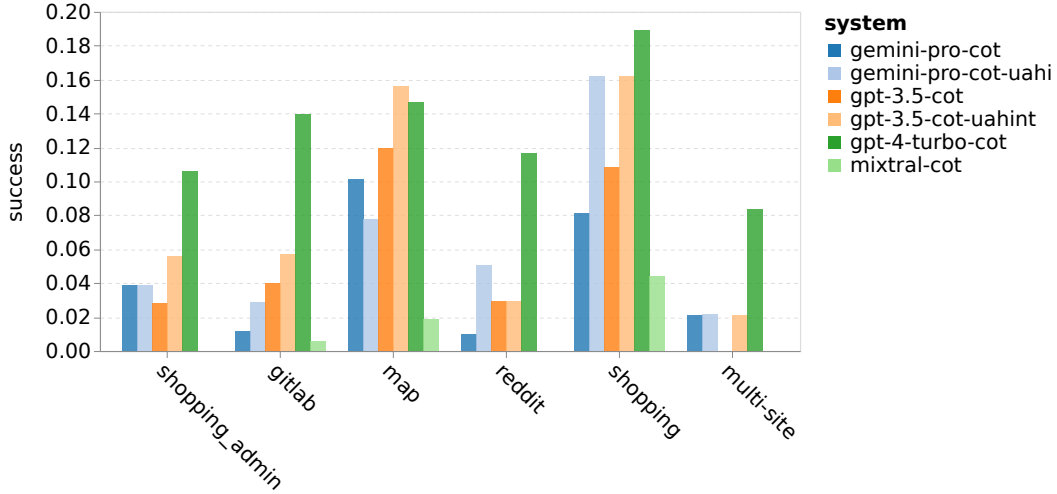


Figure 21: Web agent success rate of evaluateioned models at different site groups

8 Web Agents

Finally, we examine the ability of each model to act as a web navigation agent, a task that requires long-term planning and complex data understanding. We use WebArena [Zhou et al., 2023b], an execution-based simulation environment where the success criterion is based on execution outcome. Tasks given to agents consist of information seeking, site navigation, and content & configuration operations. The tasks span over a variety of web sites, including E-commerce platforms, social forums, collaborative software development platforms (e.g. gitlab), content management systems, and online maps.

CoT	UA Hint	Model	SR	SR _{AC}
✓	✓	GEMINI-PRO	7.09	3.52
✓	✗	GEMINI-PRO	5.23	4.83
✓	✓	GPT-3.5-TURBO	8.75	6.44
✓	✗	GPT-3.5-TURBO	6.41	6.06
✓	✗	GPT-4-TURBO	15.16	14.22

Table 6: Performances on WebArena.

8.1 Experiment Details

Generation Parameters We follow WebArena’s testing methodology in testing Gemini. We used the two-shot chain-of-thought prompts from Zhou et al. [2023b], where each prompt includes two CoT style examples. We further distinguished between whether or not the model is instructed to terminate execution when it believes the task is unachievable (the “unachievable” hint, or UA in WebArena parlance).

In sum, we tested with two prompts from WebArena: `p_cot_id_actree_2s` and `p_cot_id_actree_2s_no_na`, which are respectively CoT prompt with the UA hint and CoT prompt without the UA hint. To make results comparable between GPTs and Gemini, we set the same upper limit on the observation lengths for all of them. This number is set to 1920 tokens using the tokenizer of `gpt-4-1106-preview`, consistent with experiments in WebArena. In terms of hyper-parameters, we used the default suggested by each of the large language model providers. For the Gemini models, the suggested default temperature is 0.9 and default top-p is 1.0, and the WebArena’s suggested default for GPT models is 1.0 for temperature and 0.9 for top-p.

Evaluation Procedure The action sequence of an agent is considered correct as long as they achieved the final goal, regardless the intermediate steps they take. We use WebArena’s evaluation, which determines whether a task is completed successfully or not with the agent’s final output. A small number of responses were blocked by the Gemini API (around 2% of the total test cases), and we treat these as failed trajectories in our experiments.

8.2 Results and Analysis

We examine Gemini-Pro’s overall success rate, rate across different tasks, its response lengths, trajectory step counts, and tendency to predict that the task is unachievable. The overall performance is list in Table 6. Gemini-Pro performs comparably but slightly worse than GPT-3.5-Turbo. Similarly to GPT-3.5-Turbo, Gemini-Pro performs better when the prompt mentions that task might be unachievable (UA hint). With UA hint, Gemini-Pro achieves an overall 7.09 percent success rate.

If we break down by websites, as shown in Figure 21, we can see that Gemini-Pro performs worse than GPT-3.5-Turbo on gitlab and maps, while being close to GPT-3.5-Turbo on shopping admin, reddit, and shopping. It performs better than GPT-3.5-Turbo on multi-site tasks, which is in concert with our previous results of Gemini being a bit better on the more complex sub-tasks across benchmarks.

In general, Gemini-Pro predicts more tasks as unachievable, especially in the case where a UA hint is given, as shown in Figure 22. Gemini-Pro predicts over 80.6% of the tasks as unachievable when given an UA hint, compared to 47.7% by GPT-3.5-Turbo. Note that 4.4% of the tasks in the dataset are actually unachievable, so both far over-predict the actual number of unachievable tasks.

At the same time, we observed that Gemini Pro has a higher tendency to respond in shorter phrases and take fewer steps before reaching a conclusion. As shown in Figure 23a, more than half of trajectories by Gemini Pro are under ten steps, while majority of trajectories by GPT 3.5 Turbo and GPT 4 Turbo are between 10 and 30 steps. Similarly, the majority of Gemini responses are less than 100 characters in length, while most of GPT 3.5 Turbo, GPT 4 Turbo, and Mixtral’s responses are

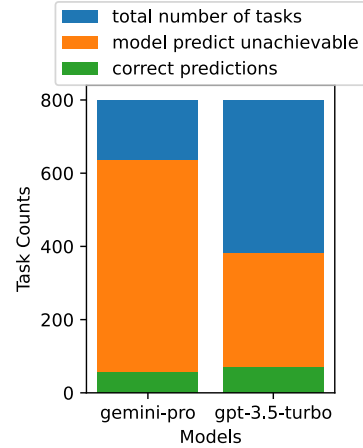


Figure 22: UA prediction count

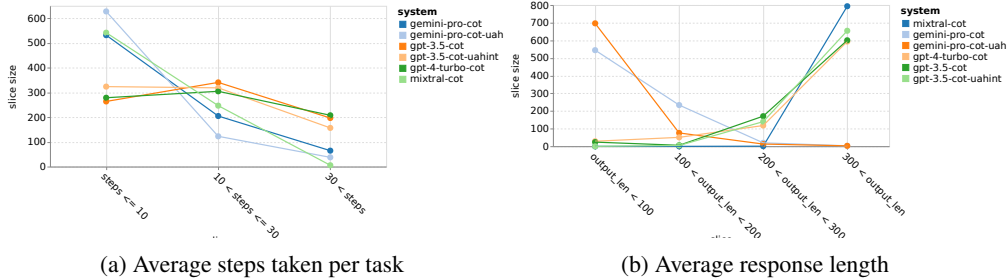


Figure 23: Model behaviors on WebArena.

over 300 characters in length [Figure 23b](#). Gemini tends to directly predict the actions while other models would start with reasoning and then give their action predictions.

9 Conclusion

In this paper, we have taken a first *impartial, in-depth* look into Google’s Gemini model, comparing it to OpenAI’s GPT 3.5 and 4 models, as well as the open source Mixtral model.

Takeaways We came away with a number of conclusions:

- The Gemini Pro model, which is comparable to GPT 3.5 Turbo in model size and class, generally achieves accuracy that is comparable but somewhat inferior to GPT 3.5 Turbo, and much worse than GPT 4. It outperforms Mixtral on every task that we examined.
- In particular, we find that Gemini Pro was somewhat less performant than GPT 3.5 Turbo on average, but in particular had issues of bias to response order in multiple-choice questions, mathematical reasoning with large digits, premature termination of agentic tasks, as well as failed responses due to aggressive content filtering.
- On the other hand, there were bright points: Gemini performed better than GPT 3.5 Turbo on particularly long and complex reasoning tasks, and also was adept multilingually in tasks where responses were not filtered.

Limitations Finally, we would like to temper these conclusions with a number of limitations.

First, our work is a snapshot in time with respect to ever-changing and unstable API-based systems. All results here are current as of this writing on December 19, 2023, but may change in the future as models and the surrounding systems are upgraded.

Second, the results may be dependent on the specific prompts and generation parameters that we selected. It is quite possible that with further prompt engineering, or multiple samples and self-consistency as was used by [Gemini Team \[2023\]](#), the results could change significantly. However, we believe that the consistent results over several tasks with standardized prompts is a reasonable indication of the robustness and generalized instruction following capability of the tested models.

Finally, any benchmarking paper would be remiss without a discussion of data leakage, which plagues current evaluation of large language models [[Zhou et al., 2023a](#)]. While we did not measure this leakage explicitly, we did attempt to mitigate by evaluating on a broad variety of tasks, including those whose outputs were not sourced from or widely available on the internet (such as WebArena).

Outlook Based on this paper, we can make the recommendation to researchers and practitioners to carefully look at the Gemini Pro model as a tool in the toolbox, comparable to GPT 3.5 Turbo. Gemini’s Ultra edition, which is yet to be released, is reported to be on par with GPT 4, and a further examination of this model will be warranted when it is available.

Acknowledgements

The authors would like to thank Zhiruo Wang for her help in handling the ODEX dataset, and Shuyan Zhou for high-level guidance on the WebArena experiments.

References

- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I Hong, and Adam Perer. Zeno: An interactive framework for behavioral evaluation of machine learning. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2023.

- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. Chain-of-thought hub: A continuous effort to measure large language models’ reasoning performance. *arXiv preprint arXiv:2305.17306*, 2023.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023a. URL <https://zenodo.org/records/10256836>.
- Yuan Gao, Ruili Wang, and Feng Hou. How to design translation prompts for chatgpt: An empirical study. *arXiv preprint arXiv: 2304.02182*, 2023b.
- Gemini Team. Gemini: A family of highly capable multimodal models. Technical report, Google, 12 2023. URL https://storage.googleapis.com/deepmind-media/gemini/gemini_1_report.pdf.
- Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc’Aurelio Ranzato, Francisco Guzmán, and Angela Fan. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538, 2022.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.
- Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213, 2022.
- Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1136. URL <https://aclanthology.org/N16-1136>.
- Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, 2020.
- Mistral AI team. Mixtral of experts, December 2023. URL <https://mistral.ai/news/mixtral-of-experts/>. Accessed: 2023-12-15.
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit,

- Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No language left behind: Scaling human-centered machine translation. *META*, 2022.
- OpenAI. Gpt-4 technical report, 2023.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 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, 2002.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL <https://aclanthology.org/2021.naacl-main.168>.
- Maja Popović. chrF++: words helping character n-grams. In *Proceedings of the second conference on machine translation*, pages 612–618, 2017.
- Matt Post. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*, 2018.
- Raf. What are tokens and how to count them? <https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them>, 2023. Accessed: 2023-12-15.
- Nathaniel R. Robinson, Perez Ogayo, David R. Mortensen, and Graham Neubig. Chatgpt mt: Competitive for high- (but not low-) resource languages. *Conference on Machine Translation*, 2023. doi: 10.48550/arXiv.2309.07423.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.
- Lindia Tjauatja, Valerie Chen, Sherry Tongshuang Wu, Ameet Talwalkar, and Graham Neubig. Do llms exhibit human-like response biases? a case study in survey design. *arXiv preprint arXiv:2311.04076*, 2023.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022a.
- Zihuro Wang, Shuyan Zhou, Daniel Fried, and Graham Neubig. Execution-based evaluation for open-domain code generation. *arXiv preprint arXiv:2212.10481*, 2022b.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
- Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. Don’t make your llm an evaluation benchmark cheater. *arXiv preprint arXiv:2311.01964*, 2023a.
- Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023b.

A Author Contributions

Syeda Akter performed experiments, analysis, and writing for the text understanding and mathematical reasoning tasks. Zichun Yu performed experiments, analysis, and writing for the knowledge-based question answering and the code generation tasks. Aashiq Muhamed performed experiments, analysis, and writing for the machine translation task. Tianyue Ou performed experiments, analysis, and writing for the instruction following agents task. Ángel Alexander Cabrera and Alex Bäuerle provided visualization support and performed fine-grained analysis of the results for each tasks. Krrish Dholakia provided support implementing calls to each of the language models. Chenyan Xiong provided direction on the varieties of tasks to pursue and helped with paper writing. Graham Neubig proposed the project idea, wrote the introduction, experimental setup, and conclusions section, and provided analysis and writing support for all other sections.

B Prompt Details

In this section, we detail the prompts that we used for each task.

For Knowledge-based QA task in Section 3, we have used standard 5-shot prompts from Hendrycks et al. [2021]¹⁰ and 5-shot chain-of-thought prompts from chain-of-thought-hub¹¹.

For General-purpose Reasoning task in Section 4, we have used Chain-of-Thought prompts from Gao et al. [2023a]¹².

For Mathematics tasks in Section 5, we also have followed Chain-of-Thought prompts from Gao et al. [2023a]¹³.

For Code Generation in Section 6, prompt is listed in Table 7.

Prompt
Complete the given code with no more explanation. Remember that there is a 4-space indent before the first line of your generated code. [CODE BLOCK]

Table 7: Prompts used for code generation tasks.

For Machine Translation in Section 7, prompts are listed in Table 8.

For WebArena in Section 8, we used CoT with UA (unachievable) hint¹⁴ and CoT without UA hint¹⁵.

¹⁰<https://github.com/hendrycks/test>

¹¹https://github.com/FranxYao/chain-of-thought-hub/blob/main/MMLU/lib_prompt/mmlu-cot.json

¹²https://github.com/EleutherAI/lm-evaluation-harness/tree/big-refactor/lm_eval/tasks/bbh/cot_fewshot

¹³https://github.com/EleutherAI/lm-evaluation-harness/blob/big-refactor/lm_eval/tasks/gsm8k/gsm8k-cot.yaml

¹⁴https://github.com/ooottyyy/webarena/blob/main/agent/prompts/raw/p_cot_id_actree_2s.py

¹⁵https://github.com/ooottyyy/webarena/blob/main/agent/prompts/raw/p_cot_id_actree_2s_no_na.py

Shot	Prompt
zero	<p>This is an English to [TGT] translation, please provide the [TGT] translation for this sentence. Do not provide any explanations or text apart from the translation.</p> <p>[SRC]: [src-sentence] [TGT]:</p>
five	<p>This is an English to [TGT] translation, please provide the [TGT] translation for these sentences:</p> <p>[SRC]: [src-sentence] [TGT]: [tgt-sentence] [SRC]: [src-sentence] [TGT]: [tgt-sentence] [SRC]: [src-sentence] [TGT]: [tgt-sentence] [SRC]: [src-sentence] [TGT]: [tgt-sentence] [SRC]: [src-sentence] [TGT]: [tgt-sentence]</p> <p>Please provide the translation for the following sentence. Do not provide any explanations or text apart from the translation.</p> <p>[SRC]: [src-sentence] [TGT]:</p>

Table 8: Prompts used for zero- and five-shot settings in translation tasks.