Does Prompt Formatting Have Any Impact on LLM Performance?

Anonymous ACL submission

Abstract

In the realm of Large Language Models (LLMs), prompt optimization is crucial for model performance. Although previous research has explored aspects like rephrasing prompt contexts, using various prompting techniques (like in-context learning and chain-of-thought), and ordering few-shot examples, our understanding of LLM sensitivity to prompt templates remains limited. Therefore, this paper examines the impact of different prompt templates on LLM performance. We formatted the same contexts into various human-readable templates, including plain text, Markdown, JSON, and YAML, and evaluated their impact across tasks like natural language reasoning, code generation, and translation using OpenAI's GPT models. Experiments show that GPT-3.5-turbo's performance varies by up to 40% in a code translation task depending on the prompt template, while larger models like GPT-4 are more robust to these variations. Our analysis highlights the need to reconsider the use of fixed prompt templates, as different formats can significantly affect model performance.

1 Introduction

The emergence of LLMs marks a significant advancement in AI, revolutionizing natural language processing, understanding, and generation ((Brown et al., 2020; Ouyang et al., 2022; Chowdhery et al., 2022; Achiam et al., 2023)). Prompt engineering has become crucial, focusing on crafting inputs that guide LLMs to produce desired outputs, leveraging a nuanced understanding of how these models interpret and respond to prompts ((Sahoo et al., 2024)). Effective prompt design generally includes clear instructions, Retrieval-Augmented Generation (RAG) or other prompting approaches for enhancing in-context learning (ICL), and appropriate formatting.

Often overlooked, prompt format can significantly impact model performance, contrary to the assumption that it remains stable across different templates. There exist limited research and anecdotal evidence ((Aghajanyan, June 2023; Sclar et al., 2023; Voronov et al., 2024)), which suggest that prompt format choices may lead to substantial performance variations, raising concerns about current evaluation standards that ignore this factor. For example, one study showed that LLMs are sensitive to minor fine-grained prompt modifications, such as separators or capitalization changes ((Sclar et al., 2023)). Also, existing evaluation approaches typically use fixed templates, potentially leading to misleading conclusions ((Voronov et al., 2024)).

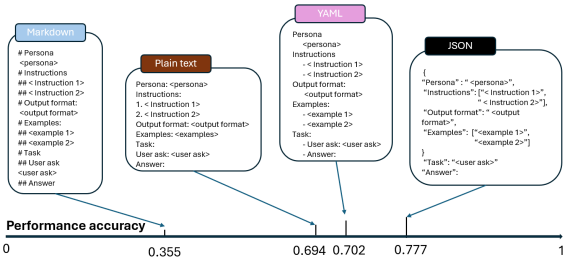


Figure 1: An example to demonstrate how prompt formatting impacts GPT-35-turbo-16k-0613 model's performance based on our experiments on multiple choice questions related to international law from the MMLU benchmark ((Hendrycks et al., 2020)). Texts inside "<>" are replaced by actual contexts. Accuracy goes up by 42% for JSON compared to Markdown.

Inspired by these findings, our study investigates whether broader changes in prompt format affect model efficacy. We evaluate the impact of prompt templates on OpenAI's four GPT models across six benchmarks, using plain text, Markdown, YAML, and JSON formats, as illustrated in Figure 1. This comprehensive approach contrasts with prior research that primarily examined minor template alterations. Our research focuses on the GPT model series for two main reasons: the lack of comparative analyses of behavioral patterns across different

GPT model iterations, especially the latest GPT-4-turbo, and the need to identify effective interaction methods and optimal input formats for these models, which do not disclose their training methodologies or data.

Our study is designed to investigate the following key questions:

- **Sensitivity**: To what extent does the performance of GPT models vary with different prompt formats?
- **Consistency**: Are GPT models capable of producing uniform responses to identical queries when presented with varying prompt structures?
- **Transferability**: Is there an optimal prompt format that is universally effective across diverse GPT models, thereby ensuring peak performance?

In addition to our primary questions, we explore the correlation between prompt format efficacy and task-specific competencies, as well as the impact of model size on performance. OpenAI's GPT models including GPT-35-turbo and GPT-4 (Achiam et al., 2023) show unpredictable sensitivity to prompt format changes, with significant performance discrepancies across all models and benchmarks. Notably, there is no universally optimal format, even within the same generational lineage. However, GPT-4-turbo demonstrates greater resilience to prompt format changes compared to its predecessors and contemporaries. In summary, our key contributions are as follows:

- This study is the first to compare the impact of different prompt formats on GPT models' performance across various tasks, examining plain text, Markdown, YAML, and JSON.
- Our research provides an extensive analysis of prompt formatting effects on GPT models across a wide range of tasks, including multiple-choice questions, code generation, and translation.
- We present an evaluation of the GPT model iterations via Azure OpenAI, revealing that GPT-4-turbo is less susceptible to prompt structure variations compared to earlier models.

2 Experimental Setup

2.1 Datasets

Our experiments span various tasks and datasets, categorized into three main groups:

- **Natural Language to Natural Language (NL2NL)**: Includes **Massive Multitask Language Understanding (MMLU)** (Hendrycks et al., 2020) and **NER Finance** from OpenAI Evals (OpenAI, 2023).
- **Natural Language to Code (NL2Code)**: Includes **HumanEval** (Chen et al., 2021) and **FIND** (Schwettmann et al., 2023).
- **Code to Code (Code2Code)**: Includes **CODEXGLUE** (Lu et al., 2021) and **HumanEval-X** (Zheng et al., 2023).

We initially assess model performance using task-specific scalar scoring functions, followed by metrics from Sections 3 to 5 to address our research questions. Detailed dataset descriptions and metrics are in Appendix B.

2.2 Prompt Design

We use various input formats: plain text, markdown, YAML, and JSON. Prompts include five components: *persona*, *task instructions*, *examples*, *output format instructions*, and *user ask*. We ensure the content of each placeholder stays the same across different prompt formats. The only differences are in structure and syntax. To avoid confounding variables, we design the prompts so that the context and meaning remain consistent, regardless of the format. Examples are in Appendix C.

2.3 Models

Experiments were conducted on OpenAI's GPT-3.5 and GPT-4 models via Azure (Microsoft, 2024). For GPT-3.5, we used "gpt-35-turbo-0613" and "gpt-35-turbo-16k-0613" to compare context window sizes (4k vs. 16k). For GPT-4, we used "gpt-4-32k-0613" and "gpt-4-1106-preview" to test the newer, faster variant with a 128k context window.

3 Sensitivity

3.1 Metrics Definition

Sensitivity. To evaluate how much the choice of prompt template impacts a model's performance on a task $\mathbf{T}$, we look at a variety of templates $\{p_1, p_2, \ldots, p_n\}$ and measure their performance

	GPT-35-turbo-0613			GPT-35-turbo-16k-0613			GPT-4-1106-preview			GPT-4-32k-0613		
	Max	Min	p-value	Max	Min	p-value	Max	Min	p-value	Max	Min	p-value
MMLU	59.7 (JSON)	50.0 (Markdown)	< 0.001	59.4 (JSON)	50.7 (Markdown)	< 0.001	81.2 (Markdown)	73.9 (JSON)	< 0.001	81.3 (Markdown)	77.8 (JSON)	< 0.001
HumanEval	59.8 (JSON)	40.2 (Plain text)	< 0.001	57.9 (JSON)	37.20 (Plain text)	< 0.001	86.6 (Markdown)	82.9 (Plain text)	0.055	76.2 (Plain text)	21.95 (JSON)	< 0.001
NER Finance	37.2 (Plain text)	24.6 (YAML)	< 0.001	36.80 (Plain text)	21.8 (YAML)	< 0.001	53.8 (YAML)	49.3 (Plain text)	0.001	53.2 (YAML)	47.2 (Plain text)	< 0.001
CODEXGLUE (Java2CS)	78.4 (JSON)	66.5 (Plain text)	< 0.001	78.4 (JSON)	66.5 (Plain text)	< 0.001	77.0 (Markdown)	68.2 (Plain text)	< 0.001	74.2 (Markdown)	67.2 (Plain text)	< 0.001
CODEXGLUE (Cs2Java)	77.5 (JSON)	68.8 (Plain text)	< 0.001	77.5 (JSON)	68.9 (Plain text)	< 0.001	83.1 (JSON)	68.1 (Plain text)	< 0.001	75.0 (JSON)	67.9 (Plain text)	< 0.001
HumanEval-X	69.8 (YAML)	63.0 (Plain text)	< 0.001	69.8 (YAML)	62.9 (Plain text)	< 0.001	72.4 (JSON)	63.9 (Plain text)	< 0.001	72.3 (JSON)	65.0 (Plain text)	< 0.001
FIND	15.9 (Plain text)	5.2 (Markdown)	< 0.001	15.8 (Plain text)	5.0 (Markdown)	< 0.001	20.7 (Markdown)	20.08 (Plain text)	< 0.0269	21.9 (plaintext)	17.4 (markdown)	< 0.001

Table 1: Sensitivity of model performance to prompt format assessed using one-sided matched pair t-tests. Table displays metrics for top and bottom formats (Max/Min) and p-values for each dataset/model. All p-values are below 0.05, except for GPT-4-1104-preview on HumanEval, confirming widespread prompt format sensitivity.

using a scoring function s. We identify the highest score $Max(s(p_i, \mathbf{T}))$ and the lowest score $Min(s(p_i, \mathbf{T}))$ achieved by the templates for task $\mathbf{T}$. Then, we use a matched pairs t-test to determine if the performance difference is statistically significant. If the test shows significance, the prompt format matters; if not, the model's performance is relatively insensitive to the prompt used.

3.2 Does prompt format impact the performance of language models and how significant is the performance variation when switching prompt formats?

We begin by analyzing if model performance is sensitive to any changes in the prompt format at all. To assess this, we conducted a one-sided matched pair t-test, comparing the best and worst performing formats for each model across various benchmarks. The resulting p-values, which are shown in Table 1, are mostly below 0.01. This suggests that the differences in model performance due to format changes are statistically significant.

Figure 4 visualizes how the models fare across all benchmarks, highlighting a considerable range in performance. For instance, in the FIND dataset, both GPT-35-turbo-0613 and GPT-35-turbo-16k-0613 show a dramatic 200% improvement when prompts are switched from Markdown to plain text. Similarly, for the HumanEval benchmark, the GPT-4 model with a 32k-0613 configuration exhibits an impressive performance boost of over 300% when the prompt format is changed from JSON to plain text. **This suggests, LLM performance may not be robust to the choice of prompt format.**

4 Consistency

4.1 Metrics Definition

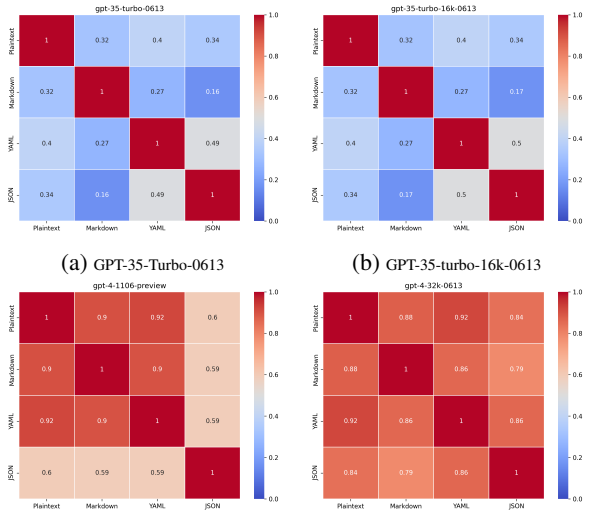
Following the sensitivity measurement, we quantify the extent of answer variation due to prompt changes using the *consistency* metric from (Shu et al., 2023). This metric calculates the proportion of test samples that yield identical responses for two prompt templates. The consistency $C(P_a, P_b)$ for templates P_a and P_b is defined as:

$$C(P_a, P_b) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1} \left(A_{P_a}(x_i) = A_{P_b}(x_i) \right)$$

where N is the test set size and A represents the model's answer. A higher score indicates greater answer consistency between prompts.

4.2 Are larger models more consistent in generated outputs between templates?

Our study assessed the consistency of model outputs using the MMLU and FIND datasets, as shown in Figures 2 and 8. For MMLU, we set the temperature to zero to eliminate response variability. The GPT-3.5-turbo series displayed low consistency, with scores below 0.5, and only 16% identical responses between Markdown and JSON formats. In contrast, GPT-4's consistency scores surpassed 0.5, indicating better reliability across different prompts. For the FIND dataset, following the settings from (Schwettmann et al., 2023), GPT-4 again outperformed the GPT-3.5-turbo series in consistency. These findings suggest that larger models like GPT-4 are more consistent, but there is still a need for model improvements to achieve reliable



(a) GPT-35-Turbo-0613 (b) GPT-35-turbo-16k-0613

(c) GPT-4-1106-preview (d) GPT-4-32k-0613

Figure 2: Consistency comparison for MMLU dataset: GPT-3.5 models show consistency scores below 0.5 across format pairs, whereas GPT-4 consistently exceeds 0.5, indicating greater reliability.

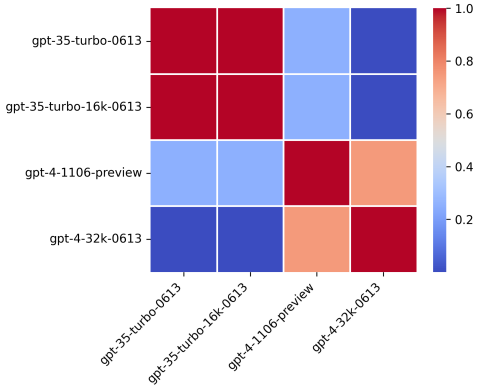


Figure 3: Intersection over Union (IoU) scores for top templates on the NER Finance benchmark across models. Higher IoU is observed within same-version model pairs, whereas cross-version pairs exhibit lower IoU.

performance across various formats. **In summary, the consistency of model responses varies with size, with larger models like GPT-4 providing more uniform outputs across different prompts.**

5 Transferability

5.1 Metrics Definition

Intersection-over-Union. To assess the transferability of prompt templates between models, we calculate the Intersection-over-Union (IoU) for the sets of top-performing templates between model pairs. Top-performing templates are those with statistically indistinguishable performance, determined by a matched pairs t-test. The IoU is defined as:

$$\text{IoU} = \frac{|P_{m1} \cap P_{m2}|}{|P_{m1} \cup P_{m2}|}$$

where P_{m1} and P_{m2} represent the sets of top templates for models $m1$ and $m2$, respectively. An IoU threshold of 0.5 is common, but a higher threshold like 0.7 indicates greater overlap.

5.2 Do models from same family exhibit similar trend across prompt formats?

Our research into Large Language Models (LLMs), GPT-based models in particular, reveals that prompt formatting preferences vary by model. As demonstrated in Figure 5, GPT-3.5-turbo prefers JSON, whereas GPT-4 favors Markdown. When examining prompt transferability using Intersection-over-Union (IoU) metrics (Figure 3 and Appendix B), we found low compatibility between different model series, with IoU often below 0.2. However, models from the same sub-series, like GPT-35-turbo-16k-0613 and GPT-35-turbo-0613, show high IoU over 0.7.

These insights highlight that even with common architectures and training goals, GPT-models react differently to identical prompts. Optimal performance requires model-specific prompt engineering, as no single format works universally across various GPT models, even within the same family. **This underscores the necessity for tailored prompt engineering due to the non-transferability of prompt formats across different GPT models.**

6 Conclusion

Our study reveals that the way prompts are formatted significantly impacts GPT-based models' performance, with no single format excelling universally. This finding questions current evaluation methods that often ignore prompt structure, potentially misjudging a model's true abilities. We advocate for diverse prompt formats in future LLM testing to accurately gauge and enhance their performance.

Regarding explainability, we observe that model size affects model's responses to prompt variations. For instance, GPT-4's performance is less influenced by prompt changes compared to GPT-3.5, suggesting that larger models may process prompts more consistently. This discovery prompts further research into LLM interpretability, aiming to refine AI adaptability and human-AI interaction.

7 Limitations

This study was focused on GPT-based models, however, we plan to examine the impact of prompt formats on other models, such as LLaMA (Touvron et al., 2023), Gemini (Team et al., 2023), PaLM (Chowdhery et al., 2022), or smaller models like Phi (Li et al., 2023) in the future. This would provide a more holistic understanding of the influence that prompt formatting exerts across different LLM families.

Moreover, there is an opportunity to enhance the breadth of template exploration in subsequent studies. Our research did not include formats like HTML or XML, which are prevalent in the training datasets of many models. Incorporating these formats could yield a more exhaustive examination of prompt format effects.

Lastly, our experimental design maintained all other prompt design elements constant, isolating prompt format as the sole variable. It would be intriguing for future work to investigate how the sensitivity of models to prompt format might shift when other prompt engineering techniques are modified. This includes varying the number of few-shot examples provided or refining the precision of prompt instructions. Such research could offer valuable insights into the interplay between prompt structure and model responsiveness, potentially informing more effective prompt engineering practices.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Armen Aghajanyan. June 2023. Tweet: Susan and i found mmlu performance jump 6-10 points in the 40s by formatting multiple choice as (a) not a in mmlu (for internal model). all evaluation of llm's are broken. evaluating a task requires marginalizing across all prompts that describe the task, not point estimate of one.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. arxiv. *arXiv preprint arXiv:2303.12712*.

Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, et al. 2024. Stealing part of a production language model. *arXiv preprint arXiv:2403.06634*.

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. 2021. Evaluating large language models trained on code.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, et al. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *Preprint*, arXiv:2104.08786.

Microsoft. guidance. https://github.com/guidance-ai.

Microsoft. 2024. Azure openai service models. `https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/models#gpt-4-and-gpt-4-turbo-preview`. Accessed: 2024-03-26.

OpenAI. 2023. Evals. `https://github.com/openai/evals`.

OpenAI. 2024. New embedding models and api updates. Accessed: 2024-03-26.

OpenAI. November 2023. Improved instruction following and json mode.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

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.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *Preprint*, arXiv:2402.07927.

Sarah Schwettmann, Tamar Rott Shaham, Joanna Materzynska, Neil Chowdhury, Shuang Li, Jacob Andreas, David Bau, and Antonio Torralba. 2023. Find: A function description benchmark for evaluating interpretability methods. *Preprint*, arXiv:2309.03886.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*.

Bangzhao Shu, Lechen Zhang, Minje Choi, Lavinia Dunagan, Dallas Card, and David Jurgens. 2023. You don't need a personality test to know these models are unreliable: Assessing the reliability of large language models on psychometric instruments. *arXiv preprint arXiv:2311.09718*.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *The 17th ACM International Conference on Web Search and Data Mining (WSDM '24)*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Anton Voronov, Lena Wolf, and Max Ryabinin. 2024. Mind your format: Towards consistent evaluation of in-context learning improvements. *arXiv preprint arXiv:2401.06766*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *Preprint*, arXiv:2210.03493.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *Preprint*, arXiv:2102.09690.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. 2023. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x. *Preprint*, arXiv:2303.17568.

A Related Work

Prompt Engineering The field of prompt engineering has garnered significant interest in recent years, in parts due to the emergent capabilities of the most capable LLMs, while also trying to better control their still unpredictable outcomes. A prominent strand of research within this domain concentrates on innovative prompting methodologies. These include few-shot prompting ((Brown et al., 2020)), which enables models to adapt to new tasks without extensive retraining, and Chain-of-Thought prompting ((Wei et al., 2023)), both of which are designed to enhance the reasoning capabilities of LLMs. Additionally, Automatic Chain-of-Thought (Auto-CoT) ((Zhang et al., 2022)) and Self-Consistency ((Wang et al., 2023)) approaches have been developed to further refine these reasoning processes. To mitigate hallucinations in LLM outputs, techniques such as Retrieval Augmented Generation (RAG) ((Lewis et al., 2021)) and ReAct ((Yao et al., 2023)) have been introduced. A thorough examination of these methodologies can be found in the survey by (Sahoo et al., 2024). In recent developments, a novel prompt programming framework ((Microsoft)) has been introduced, which offers greater control and efficiency in generating structured outputs. Our study diverges from this approach by examining the effects of more prevalent and established prompt formats on LLMs, as opposed to investigating formats that are newly proposed and not widely adopted yet. Furthermore, it is important to note that third-party tools are predominantly designed for integration with open-source models, which may not seamlessly extend to proprietary models such as GPT. Another similar vein of research is dedicated to the structural design of prompts, aiming to optimize task performance without altering the inherent semantic content. This includes investigations into the sequential arrangement of context ((Liu et al., 2023; Zhao et al., 2021; Lu et al., 2022)) and the design of prompt formats ((Sclar et al., 2023; Voronov et al., 2024; Shu et al., 2023)). Our work contributes to this growing body of literature by examining the impact of prompt formatting on the performance of LLMs.

Prompt Format The sensitivity of LLMs to prompt construction is a well-documented phenomenon, yet research on the impact of prompt formats on model performance remains sparse. Pioneering studies ((Sclar et al., 2023; Voronov et al., 2024; Shu et al., 2023)) have conducted rigorous investigations, revealing that widely used open-source LLMs exhibit extreme sensitivity to variations in prompt format. These studies, however, primarily focus on subtle, local changes to the format—such as the number of colons following a question, the insertion of newlines, or the selection of input/output verbalizers. Besides, their experimental designs are confined to classification tasks, limiting the generalizability of findings across diverse tasks.

Our research diverges from these existing studies by examining the effects of global prompt format modifications on model performance, offering insights that are applicable to a broad spectrum of LLM-based tasks that necessitate prompt engineering. The closest related work to ours is by (Sui et al., 2024), which however only provides a cursory exploration of format influence and is restricted to tabular data. To the best of our knowledge, our study is the first effort to systematically investigate the impact of global prompt format variations - an inescapable aspect of prompt engineering design decisions.

B Datasets

We evaluate six distinct benchmarks and classify them according to the nature of the task involved.

NL2NL

- **Massive Multitask Language Understanding (MMLU)** (Hendrycks et al., 2020) covers 57 subjects including 20 STEM subjects, 13 humanities subjects, 12 social sciences subjects and 12 other subjects. Each subject contains at least 100 multiple choice questions, which tests both world knowledge and problem solving ability. We use the dev set which contains 5 questions per subjects as few-shot

examples, and use test set containing 14,079 questions with different levels of difficulty to evaluation model performance. We use accuracy score to measure model performance.

- **NER Finance:** OpenAI Evals (OpenAI, 2023) is a framework containing a registry of evaluations to test LLMs where NER Finance is one of those. This dataset contains samples between one sentence to one paragraph long from financial documents. The task is to extract the all of the entities in the document, with the evaluation being if the LLM outputs each entity, in order. We randomly sample 500 examples from this dataset.

NL2Code

- **HumanEval** (Chen et al., 2021) is a benchmark dataset consisting of a collection of Python programming problems, each accompanied by a function signature, a docstring outlining the problem to be solved, and a set of unit tests that the correct implementation must pass. We use the evaluation metric pass@1, which checks if the the generated code passes the unit given unit tests in one attempt. We use all 164 samples in this dataset.

- **FIND** (Schwettmann et al., 2023): The Function Interpretation and Description (FIND) benchmark dataset is a natural language-to-code generation task. The LLM is given 5 example inputs and outputs to an unknown Python function and is tasked with reverse engineering the original Python code. We evaluate the benchmark by comparing the output of test cases on a ground truth function with the output from LLM generated functions. We use the "strings" category of functions for the task consisting of 500 functions. We provide the LLM with 5 pairs of input and output for each function. To select these examples, we randomly sample from a dataset provided by (Schwettmann et al., 2023) that contains example test strings for each function. To evaluate the generated function code, we use the string indicator metric introduced by (Schwettmann et al., 2023) that measures the number of test cases passed by the function.

Code2Code

- **CODEXGLUE** (Lu et al., 2021) stands for General Language Understanding Evaluation benchmark for CODE. It was originally introduced to address the lack of diversified benchmarks in code intelligence by providing a diverse set of tasks, including code translation. We use the parallel code for Java and C-Sharp and vice versa. We use the test set containing 1000 parallel code in Java and C-Sharp to experiment the capabilities of the LLMs in translating code from one programming language to another. The performance of the LLMs is evaluated using the BLEU (Papineni et al., 2002) score, which compares the generated code to the reference code.

- **HumanEval-X** (Zheng et al., 2023) dataset is a benchmark designed to evaluate the multilingual capabilities of code generative models. It contains 820 high-quality, human-crafted data samples, each accompanied by test cases. The dataset supports a variety of programming languages, including Python, C++, Java, JavaScript, and Go. In this we experiment with one of the above dimension of code-translation focusing on Java to Python. To accomplish this task, we combine the "declaration" and "canonical-solution" together to finally get the overall function in the respective language. "Declaration" contains the function declaration for the respective language and "canonical solution" has the human-crafted example solution for the language. Similar to CODEXGLUE, we use the BLEU (Papineni et al., 2002) score for measuring the performance.

C Prompt Templates

In this section we provide examples of the four prompt templates we used for the NER Finance task. Prompts for all other tasks followed identical formatting. Variables that are injected into the prompt are denoted by blue text wrapped in braces. For example a user ask being injected is denoted as {USER ASK}.

D Additional Research Questions

D.1 How does the format in which information is structured and presented influence the ability to solve problems that require different skill sets?

We analyze whether model's sensitivity to prompt format changes is related to the skills required to

Prompt Format	Prompt Template
Plain text	{persona} {instructions} {examples} {output format instructions} {user ask}
Markdown	## Persona {persona} ## Instructions {instructions} ## Examples {examples} ## Output Format {Output format instructions} ## User Question {user ask}
YAML	Persona - {persona} Instructions - {instructions} Examples - {examples} Output format - {output format instructions} User question - {user ask}

JSON	{
	"Persona": "{persona}",
	"Instructions": "{instructions}",
	"Examples": "{examples}",
	"Output format": "{output format instructions}"
	}
	{
	"User ask": "{user ask}"
	}

Table 2: Prompt templates considered in this paper. Placeholders are denoted with {variable name} and get replaced with task specific context.

Plaintext

```
System:
You are a annotator working for large financial data company and are tasked with extracting named entities from
financial documents who follows strict guidelines for quality and formatting. The following sentence is from a financial
document. List the named entities in the order they appear in the sentence. If an entity appears multiple times, list
it multiples times. Entities should be stated in the format NAME - TYPE where TYPE can be PERSON, ORGANIZATION, or
LOCATION. State your final answer as a comma-separated list of entities enclosed in square brackets. Example: [Bank -
ORGANIZATION, Borrower - PERSON]. If there are no entities found, state your final answer as 'No entities found'. Provide
your chain of thought first and then respond with your final answer. Here is an example: {ICL EXAMPLE INPUT} {ICL EXAMPLE SOLUTION}

User:
{INPUT}
```

solve the task using the MMLU benchmark which comprises 57 subjects, categorized into four domains: humanities, social science, STEM, and others. Each domain encompasses various disciplines, necessitating distinct skill sets and knowledge for accurate question answering.

Figure 5 breaks down the performance on MMLU dataset by domain. We observe the performance spread exists across different tasks, and it's not signified nor eliminated by specific skills required. This suggests that the model's sensitivity to prompt formatting is a general characteristic, rather than being contingent on the specific skills or reasoning abilities required by different tasks. Performance is influenced by how information is presented to it, regardless of the complexity or na-

Markdown

```
System:
## Persona
- You are a annotator working for large financial data company are tasked with extracting named entities from financial
documents who follows strict guidelines for quality and formatting.

## Instructions
- You will be given a sentence from a financial document. - List the named entities in the order they appear in the sentence.
- If an entity appears multiple times, list it multiples times.
- Provide your chain of thought first and then respond with your final answer.

## Output Format
- Entities should be stated in the format NAME - TYPE where TYPE can be PERSON, ORGANIZATION, or LOCATION.
- State your final answer as a comma-separated list of entities enclosed in square brackets. Example: [Bank - ORGANIZATION,
Borrower - PERSON].
- If there are no entities found, state your final answer as 'No entities found'.

## Example
### DOCUMENT
{ICL EXAMPLE INPUT}

### Solution
{ICL EXAMPLE SOLUTION}

User:
### DOCUMENT
{INPUT}
```

YAML

```
System:
Persona:
  Description: You are a annotator working for large financial data company are tasked with extracting named entities from
financial documents who follows strict guidelines for quality and formatting.

Instructions:
  - You will be given a sentence from a financial document.
  - List the named entities in the order they appear in the sentence.
  - If an entity appears multiple times, list it multiples times.
  - Provide your chain of thought first and then respond with your final answer.

Output_Format:
  Entities should be stated in the format NAME - TYPE where TYPE can be PERSON, ORGANIZATION, or LOCATION. State your final
answer as a comma-separated list of entities enclosed in square brackets.

Examples:
  - Document: {ICL EXAMPLE INPUT}
  - Solution: {ICL EXAMPLE SOLUTION}

User:
Task:
  - Document: {INPUT}
```

ture of the task, the way in which a problem is framed and communicated to the model can significantly impact its ability to process and respond to the information. **Model performance is consistently influenced by prompt formatting across various tasks, regardless of the specific skills or knowledge required.**

D.2 Is there a correlation between the model size and the robustness of the LLM for different prompt templates?

Coefficient of Mean Deviation. We measure the performance dispersion for a model caused by format and compare if the degree of dispersion can be attributed to the size of the model. We compute the Coefficient of Mean Deviation (CMD) across all the prompt templates for every model.

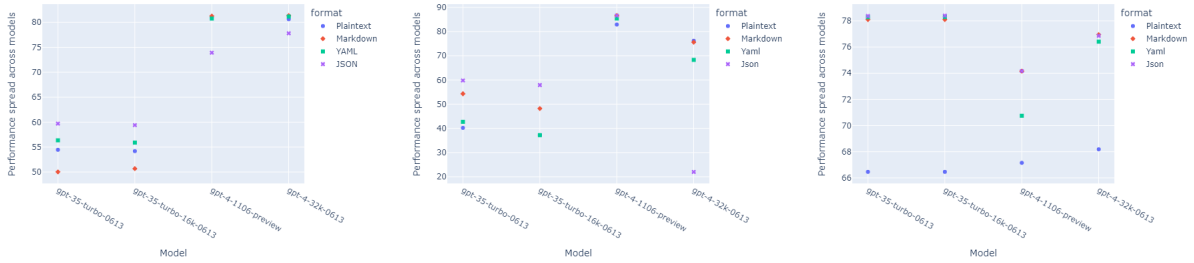
$$CMD = \frac{\sum |s(p_i) - \bar{s}|}{n \cdot \bar{s}}$$

Here $s(p_i)$ is the performance score for prompt p_i, $\bar{s}$ is the average score across all prompt formats, and n is the number of formats. A lower CMD indicates that the model's performance is less affected by prompt format changes, showing greater robustness. A higher CMD suggests more variability and

JSON

```
System:
{
  "Persona": "You are a annotator working for large financial data company are tasked with extracting named entities from financial documents who follows strict guidelines for quality and formatting.",
  "Instructions": [
    "You will be given a sentence from a financial document.",
    "List the named entities in the order they appear in the sentence.",
    "If an entity appears multiple times, list it multiples times.",
    "Provide your chain of thought first and then respond with your final answer."
  ],
  "OutputFormat": "Entities should be stated in the format NAME - TYPE where TYPE can be PERSON, ORGANIZATION, or LOCATION. State your final answer as a comma-separated list of entities enclosed in square brackets. Example: [Bank - ORGANIZATION, Borrower - PERSON]. If there are no entities found, state your final answer as 'No entities found'.",
  "Example": "{ICL EXAMPLE INPUT}\n{ICL EXAMPLE SOLUTION}"
}

User:
{
  "Task": "{INPUT}"
}
```



(a) MMLU (b) HumanEval (c) CODEXGLUE (Java2CS)

Figure 4: Model performance across prompt formats on MMLU, HumanEval and CODEXGLUE. Performance measuredment for MMLU is accuracy, for HumanEval is pass@1 to hecks if the the generated code passes the unit given unit tests in one attempt, for CODEXGLUE(Java2CS) is BLEU score. Plots for the remaining datasets are in Figure 9.

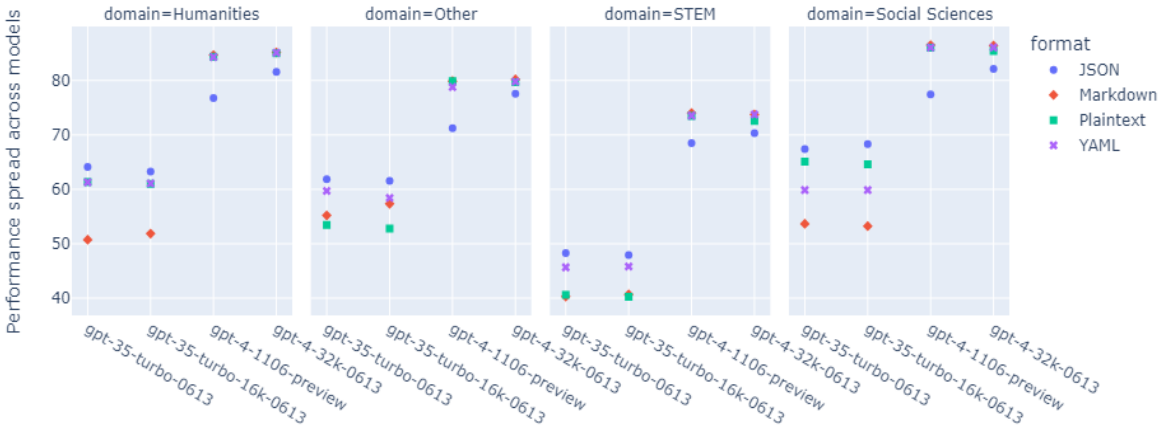


Figure 5: Performance spread across models on MMLU benchmark per domain. Wide performance spread is observed across domains that required different skills.

sensitivity to prompt format.

While the architectural details and exact size of GPT-4 are not published, it is assumed that GPT-4 contains significantly more parameters, was

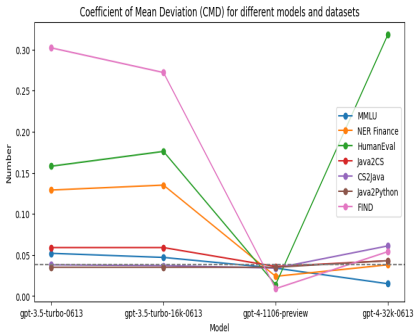


Figure 6: Coefficient of mean deviation (CMD) of scalar metrics for all the prompt templates. Figure shows the CMDs across models and datasets. GPT-3.5 series exhibit larger CMD scores across benchmarks than GPT-4 series, indicating higher sensitivity to the choice of format.

trained on more data than GPT-3.5, and is clearly the overall more capable model ((Achiam et al., 2023; Bubeck et al., 2023; Carlini et al., 2024)). In this section, we aim to ascertain whether an expansion in general capability translates to enhanced stability in response to changes in templates. The CMDs for all the models across benchmarks are presented in Figure 6.

A lower value of CMD indicates more robustness to template variation. The results indicate that the GPT-4-1106-preview model exhibits superior robustness to format changes, maintaining a performance dispersion consistently below 0.036 across all benchmarks. In contrast, the GPT-4-32k-0613 model demonstrates less robustness relative to the GPT-4-1106-preview, yet it outperforms the GPT-3.5 series, with CMDs not exceeding 0.043. The GPT-3.5 series displays a broader range of CMDs, from 0.035 to 0.176, signifying a higher degree of performance variability under different prompt formats. GPT-4's observed improvements may be attributed to its enhanced ability to process data in diverse formats. Moreover, it is possible that the robustness of the model is not adversely impacted by format variations at the level of the last hidden layer of prompt embedding. Notably, the GPT-4-1106-preview model achieves greater robustness compared to the GPT-4-32k-0613, corroborating existing evidence that suggests the former has a heightened proficiency in comprehending and generating content in specific formats as instructed (OpenAI, November 2023). Further examining GPT-4-32k-0613's performance, we notice the CMD on HumanEval benchmark is extremely high, this is due the extremely low score using JSON format, see Table 4 for results. Analyzing the model outputs, we find the poor performance is because most of the time the model would generate chain of thought in plain text, but did not continue with actually generating the code. The other models did not exhibit this behavior for the JSON template. We hypothesize that this may be related to the OpenAI's claim about fixing laziness in task completion in the 0125 version of GPT-4-turbo (OpenAI, 2024). In summary, **larger models are more robust to template variation.**

E Complete Results

E.1 Additional results on model performance under all templates across benchamrks.

E.2 IoU scores on all benchmarks.

E.3 Dotplot on all benchmark datasets

Model	GPT-35-turbo-0613	GPT-35-turbo-16k-0613	GPT-4-1106-preview	GPT-4-32k-0613
Plaintext	54.464 ± 18.300	54.184 ± 19.066	81.005 ± 12.979	80.638 ± 13.172
Markdown	50.021 ± 17.144	50.686 ± 17.436	**81.252** ± 12.932	**81.349** ± 13.158
YAML	56.355 ± 16.792	55.901 ± 16.347	80.758 ± 13.000	81.162 ± 13.110
JSON	**59.705** ± 16.594	**59.405** ± 17.092	73.918 ± 13.580	77.800 ± 13.725

Table 3: Model Performance on MMLU Benchmark. Accuracy is averaged over 57 different subjects.

Model	GPT-35-turbo-0613	GPT-35-turbo-16k-0613	GPT-4-1106-preview	GPT-4-32k-0613
Plaintext	40.24 ± 3.98	37.20 ± 3.77	82.93 ± 4.39	**76.22** ± 4.76
Markdown	54.27 ± 4.70	48.17 ± 4.44	**86.59** ± 4.06	75.61 ± 4.78
YAML	42.68 ± 4.14	37.20 ± 3.77	85.37 ± 4.18	68.29 ± 4.92
JSON	**59.76** ± 4.85	**57.93** ± 4.81	**86.59** ± 4.06	21.95 ± 2.48

Table 4: Model performance on HumanEval benchmark. We used all 164 samples for testing.

Model	GPT-35-turbo-0613	GPT-35-turbo-16k-0613	GPT-4-1106-preview	GPT-4-32k-0613
Plaintext	**37.20** ± 6.59	**36.80** ± 6.54	49.40 ± 7.86	47.20 ± 7.67
Markdown	28.00 ± 5.31	30.00 ± 5.61	51.40 ± 8.01	51.60 ± 8.03
YAML	24.60 ± 4.78	21.80 ± 4.31	**53.80** ± 8.18	**53.20** ± 8.14
JSON	28.40 ± 5.37	31.00 ± 5.76	50.80 ± 7.97	52.40 ± 8.08

Table 5: Model performance on NER finance benchmark. We randomly sampled 500 samples for testing.

Model	gpt-35-turbo-16k-0613	gpt-35-turbo-0613	gpt-4-32k-0613	gpt-4-1106-preview
plaintext	15.75±0.142	15.9±0.143	21.87±0.164	20.08±0.161
markdown	5.03±0.092	5.19±0.089	17.42±0.154	20.68±0.165
json	14.46±0.138	14.33±0.144	21.15±0.162	20.19±0.156
yaml	13.06±0.139	13.49±0.138	21.6±0.163	20.28±0.16

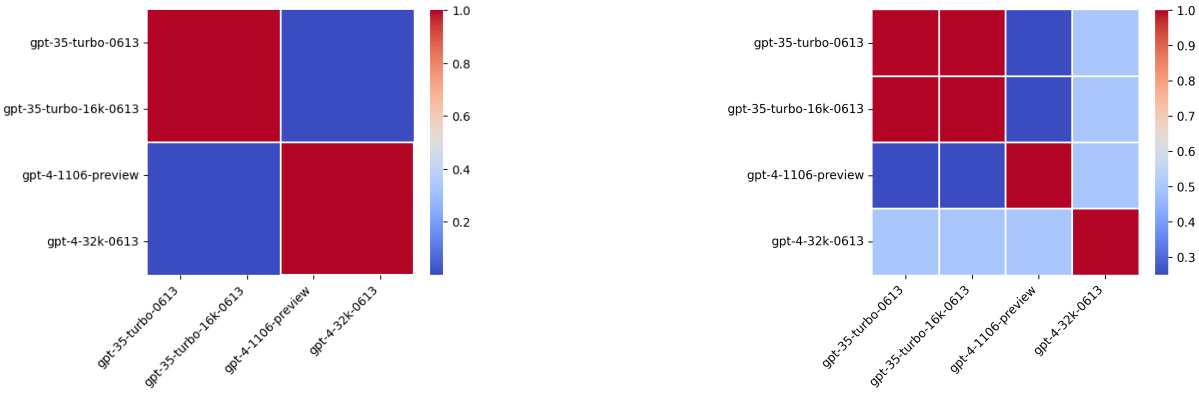
Table 6: Model performance on FIND benchmark. Test set includes 500 functions.

Model	GPT-35-turbo-0613	GPT-35-turbo-16k-0613	GPT-4-1106-preview	GPT-4-32k-0613
Plaintext	62.95 ± 15.64	62.92 ± 15.66	64.95 ± 15.52	63.86 ± 15.83
Markdown	69.82 ± 16.26	69.84 ± 16.23	70.70 ± 17.44	71.65 ± 18.10
YAML	69.05 ± 17.24	69.05 ± 17.24	71.16 ± 16.11	71.41 ± 17.96
JSON	68.85 ± 16.13	68.85 ± 16.13	72.30 ± 15.97	72.39 ± 16.46

Table 7: Model performance on HumanEval-X, a Java to Python translation task The test set contains 164 data samples.

Dataset	CODEXGLUE: Java to CS				CODEXGLUE - CS to Java			
Model	GPT-35-turbo-0613	GPT-35-turbo-16k-0613	GPT-4-1106-preview	GPT-4-32k-0613	GPT-35-turbo-0613	GPT-35-turbo-16k-0613	GPT-4-1106-preview	GPT-4-32k-0613
Plaintext	66.46 ± 16.04	66.46 ± 16.04	67.16 ± 16.77	68.19 ± 13.14	68.81 ± 17.65	68.89 ± 17.64	67.93 ± 17.72	68.11 ± 16.29
Markdown	78.10 ± 18.75	78.10 ± 18.75	74.16 ± 16.77	76.95 ± 18.33	76.19 ± 18.40	76.12 ± 18.37	74.80 ± 20.14	80.36 ± 20.52
YAML	78.28 ± 18.92	78.30 ± 18.92	70.75 ± 16.08	76.41 ± 18.00	75.47 ± 20.16	75.39 ± 20.08	72.09 ± 17.98	78.49 ± 21.23
JSON	78.37 ± 18.93	78.40 ± 18.93	74.16 ± 16.77	76.86 ± 18.31	77.49 ± 19.51	77.49 ± 19.50	75.00 ± 17.66	83.05 ± 18.60

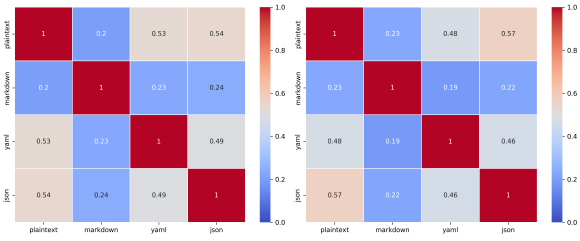
Table 8: Model performance on Java to C# and C# to Java translation task. The test set contains 1000 samples in Java and C#.



(a) MMLU

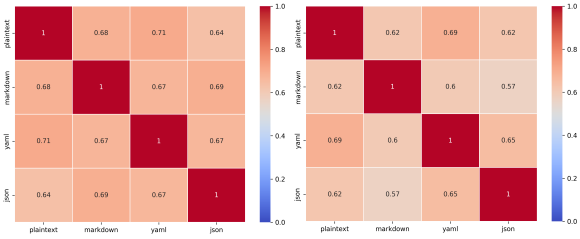
(b) FIND

Figure 7: Heatmap of IoU values for other benchmarks.



(a) GPT-35-Turbo-0613

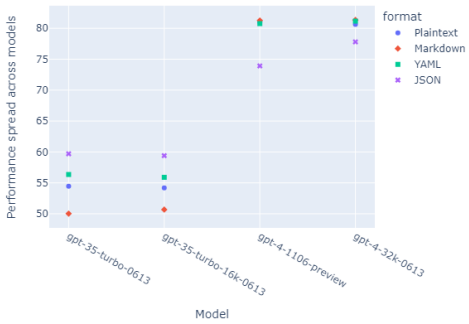
(b) GPT-35-turbo-16k-0613



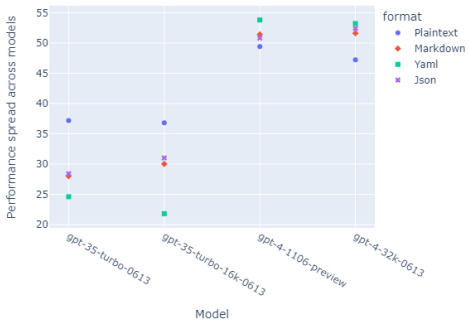
(c) GPT-4-1106-preview

(d) GPT-4-32k-0613

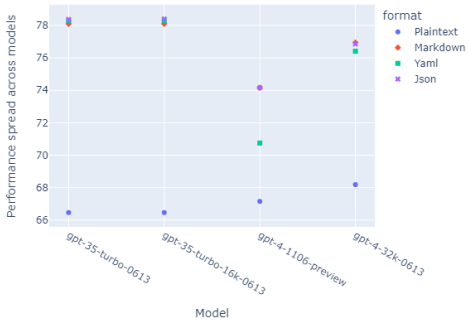
Figure 8: Performance of Consistency for FIND dataset across models.



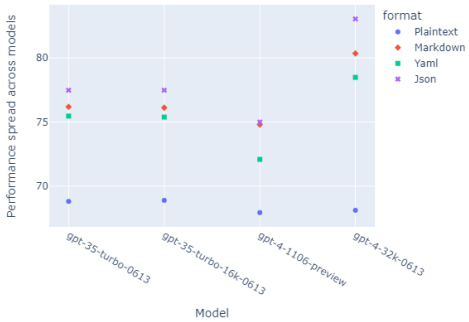
(a) MMLU



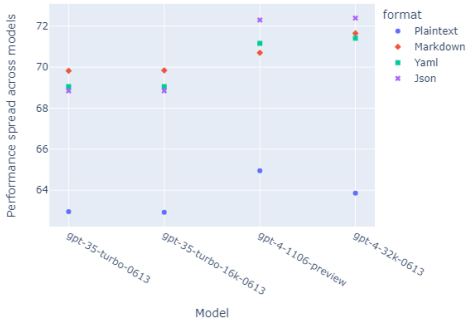
(b) NER Finance



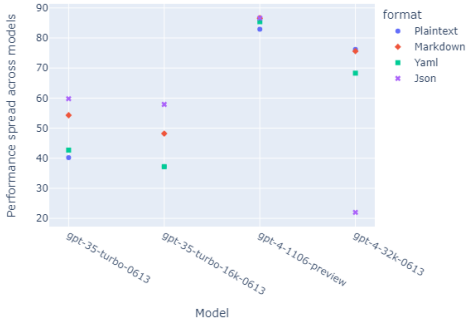
(c) CODEXGLUE (Java2CS)



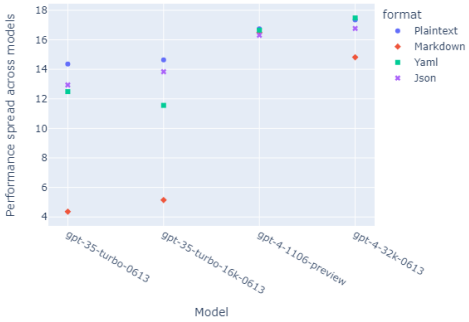
(d) CODEXGLUE (CS2Java)



(e) HumanEval-X (Java2Python)



(f) HumanEval



(g) FIND

Figure 9: Dotplot of model performance across prompt formats on all benchmarks.