Language Models Do Hard Arithmetic Tasks Easily and Hardly Do Easy Arithmetic Tasks

Andrew Gambardella * **Yusuke Iwasawa *** **Yutaka Matsuo ***
* University of Tokyo

Abstract

The ability (and inability) of large language models (LLMs) to perform arithmetic tasks has been the subject of much theoretical and practical debate. We show that LLMs are frequently able to correctly and confidently predict the first digit of n-digit by m-digit multiplication tasks without using chain of thought reasoning, despite these tasks require compounding operations to solve. Simultaneously, LLMs in practice often fail to correctly or confidently predict the last digit of an n-digit by m-digit multiplication, a task equivalent to 1-digit by 1-digit multiplication which can be easily learned or memorized. We show that the latter task can be solved more robustly when the LLM is conditioned on all of the correct higher-order digits, which on average increases the confidence of the correct last digit on 5-digit by 5-digit multiplication tasks using Llama 2-13B by over 230% (0.13→0.43) and Mistral-7B by 150% (0.22→0.55).

1 Introduction

The development of large language models (LLMs) (Brown et al., 2020) have given new life to the deep learning revolution, and seen mass adoption within not just the scientific community, but also society at large. These LLMs, being the first known "general" machine learning model developed by humanity (Morris et al., 2024), have been applied to various tasks dealing with natural language such as those commonly encountered in school curricula (Hendrycks et al., 2021), and even branching off into tasks such as text-to-image generation (Saharia et al., 2022) and hierarchical planning (Wang et al., 2023).

Despite the generality and far-reaching consequences of LLMs, there are still many significant limitations making difficult the direct application of LLMs to certain tasks. One such limitation is the poor performance of LLMs on arithmetic tasks, such as elementary addition, subtraction, multiplication, and division (Nogueira et al., 2021). Not only do modern LLMs perform poorly on these tasks, but some tasks such as n-digit by m-digit multiplication and division, which require compounding operations to solve, appear to be unlearnable by pure autoregressive transformer architectures unless they decompose the problem into multiple steps, such as with chain of thought reasoning (Wies et al., 2022; Liu et al., 2023). As such, several solutions have been proposed, such as fine-tuning so that chain of thought reasoning is automatically used for problems which require compounding operations (Liu et al., 2023; Kojima et al., 2022) or fine-tuning to call outside tools, such as a calculator (Schick et al., 2023).

While we most likely cannot expect simply training models with more parameters to allow for the solving of tasks which require compounding operations without chain of thought, we believe that analyzing the limitations and abilities of autoregressive LLMs when attempting to solve these tasks directly may shed light on unknown properties of LLMs. We therefore use Monte Carlo Dropout (MC Dropout) (Gal and Ghahramani, 2016) to analyze the performance of LLMs which were trained with dropout and which have open weights available, such as Llama 2 (Touvron et al., 2023) and Mistral (Jiang et al., 2023), in carrying out arithmetic tasks. When applicable, MC Dropout is a common and easy to use technique which can be used to obtain Bayesian confidence distributions over neural network weights or outputs, and has been applied to analyze the confidence of transformer architectures (Shelmanov et al., 2021) and to implement tree-based LLM prompting (Mo and Xin, 2023).

Our results when applying MC Dropout to Llama 2 and Mistral in arithmetic tasks were surprising. We found that all architectures could confidently and correctly predict the first digit result of

n-digit by m-digit multiplication problems, despite it most likely being impossible for any autoregressive LLM to have learned a general algorithm for doing so without decomposing the problem into multiple steps, as finding this digit in general requires solving the entire multiplication problem[1]. We also found that all architectures struggled to correctly output the last digit of n-digit by m-digit multiplication problems, despite it being very easy to learn an algorithm for doing so, as calculating the last digit is equivalent to 1-digit by 1-digit multiplication. Finally, we show that the confidence of LLMs in predicting the last digit can be increased by conditioning the generation of the last digit on the correct preceding digits, despite the computation of the last digit not depending on the correct computations of the higher-order digits.

2 Experiments

We evaluate the HuggingFace (Wolf et al., 2019) implementations of Llama 2-7B, Llama 2-13B, and Mistral-7B (Touvron et al., 2023; Jiang et al., 2023) in 2-shot settings, where the 2-shot examples are of correct n-digit by m-digit multiplications. Details about the prompt and hyperparameters are given in Appendix A. Sections 2.1 and 2.2 show results on the 3-digit by 3-digit multiplication task $592 * 392$, and averages over multiple problems with varying digit length are provided in Section 2.3.

2.1 Unconditional Answer Generation

We first study a version of the problem in which the answer is generated with the language model conditioned on the few shot examples and the problem to be solved, but is provided with none of the digits to be generated (i.e., the normal few-shot arithmetic scenario), which we refer to as "unconditional" generation in an abuse of terminology. Our main results for these experiments are in Figures 1 and 2.

In Figure 1 we can see that both Llama 2-7B and Llama 2-13B can confidently and correctly predict the first digit of the 3-digit by 3-digit multiplication task $592 * 392$, which equals 232064. This should be surprising as it is not immediately apparent from the problem that the first digit of the solution should be 2, and the only way to discover this is to compute the multiplication. As LLMs most likely cannot perform n-digit by m-digit multiplication in the general case without decomposing the problem into steps, the output of the first digit in this case is unlikely to be the output of a multiplication algorithm learned by the LLM.

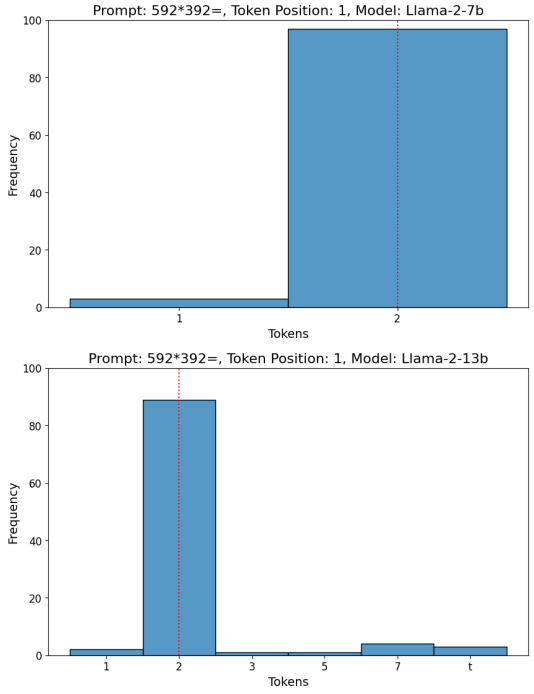

Figure 1: Confidence and accuracy of Llama 2-7B and Llama 2-13B predicting the first digit of the result of $592 * 392$. Both language models are able to confidently and correctly predict that the first digit should be 2, despite this not being immediately apparent from the problem.

Conversely, in Figure 2, we can see that both Llama 2-7B and Llama 2-13B can neither confidently nor correctly predict the last digit of the same problem, despite doing so being equivalent to 1-digit by 1-digit multiplication. This is a case in which any reasonable model should be able to confidently and correctly solve the task, as not only could the algorithm to solve the task be learned by an autoregressive language model, but the information needed to solve this task could also very easily be memorized by language models with billions of weights.

2.2 Conditional Answer Generation

Finally, we contrast the experiments given in Figures 1 and 2 with a third experiment, in which the LLM is given all digits from the answer except for the final digit, and is tasked with outputting solely the final digit, which we refer to as "conditional" generation in an abuse of terminology. Results

[1]Consider that the highest-order digit of 316227766016837933^2 is 9, but the highest-order digit of 316227766016837933^2 is 1.

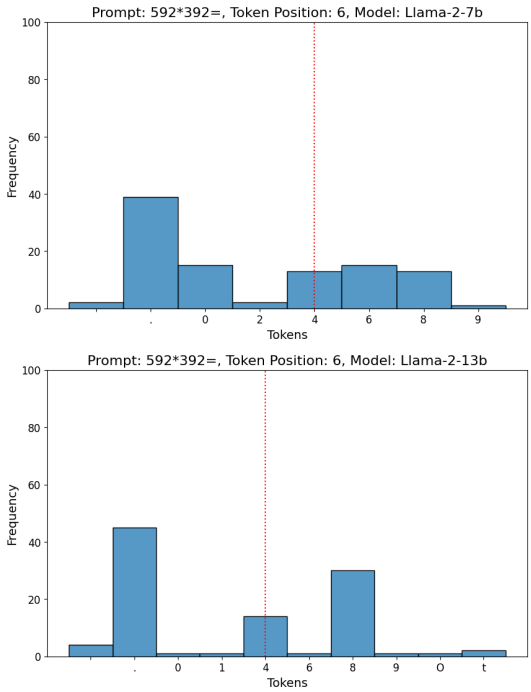

Figure 2: Confidence and accuracy of Llama 2-7B and Llama 2-13B predicting the sixth digit of the result of $592 * 392$. Neither are able to predict this digit confidently, with the mode of the distribution on the "end string" character in both cases. Both only output 4 in about 20% of samples, despite it being immediately apparent that the final digit should be 4.

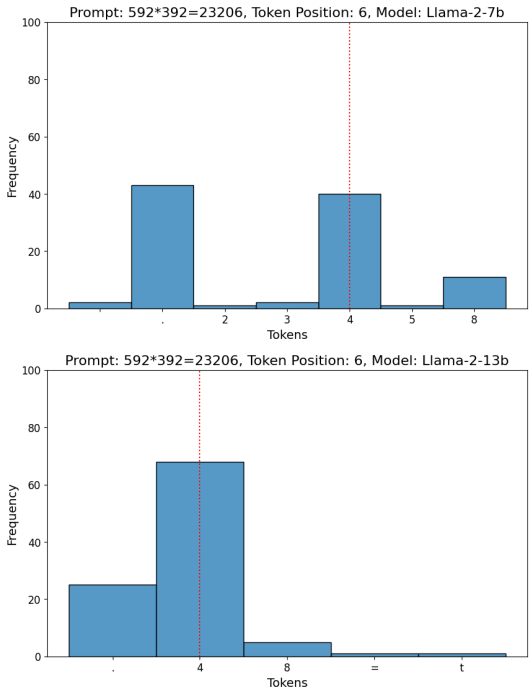

Figure 3: Confidence and accuracy of Llama 2-7B and Llama 2-13B predicting the last digit of the result of $592 * 392$, when conditioned on the first five correct digits. The confidence in the correct answer being 4 doubles for Llama 2-7B and more than triples for Llama 2-13B, despite the computation of the last digit not depending on the prior digits being correct at all.

for this experiment are given in Figure 3. In this case the confidence in the correct output doubles for Llama 2-7B and triples for Llama 2-13B, with Llama 2-13B now having most of its probability mass on the correct last digit, whereas it did not do so when generating the entire string at once (and therefore often conditioning on incorrect prior digits). The fact that in both cases, more probability mass is being put on the correct answer should be surprising, as the computation of this digit does not depend on the correctness of the higher-order digits in any way.

2.3 Ablation Over Digit Length

We provide further ablations over digit length in Table 1. Each subtable gives the confidence of the correct digit, averaged over 10 different n-digit by m-digit multiplication problems each. We find that the conclusions shown for a single example in Sections 2.1 and 2.2 hold over varying multiplication problems and digit lengths in general. We further provide Mistral-7B experiments in Appendix B where we observe the same effects.

3 Discussion of Results

3.1 First Digit

It is most likely impossible for autoregressive LLMs to compute the first digit of an n-digit by m-digit multiplication problem without decomposing the problem into steps, especially given that the answer is being written starting with the highest-order digit, and calculating the first digit depends on the correct calculations of the lower-order digits.

LLMs *can*, however, perform 1-digit by 1-digit multiplication. If these LLMs were to internally round 592 to 600 and 392 to 400, it could approximately solve for the highest-order digit in this way, as $600 * 400$ is a computation that can be performed by autoregressive language models. We find it likely that such a computation is occurring inside these LLMs, especially as stochastic gradient descent is likely to find such "shortcuts."

3.2 Last Digit

Both LLMs failing to predict the last digit when generating the entire string autoregressively, and their confidence and accuracy in predicting the last

Llama 2-7B

n \ m	2	3	4	5
2	0.81	0.90	0.82	0.82
3	0.91	0.78	0.88	0.92
4	0.88	0.83	0.92	0.77
5	0.89	0.74	0.89	0.87

(a)

Llama 2-13B

n \ m	2	3	4	5
2	0.84	0.85	0.79	0.73
3	0.87	0.72	0.85	0.86
4	0.84	0.83	0.78	0.78
5	0.86	0.71	0.84	0.86

(b)

n \ m	2	3	4	5
2	0.52	0.34	0.16	0.20
3	0.39	0.22	0.16	0.19
4	0.40	0.21	0.20	0.15
5	0.33	0.20	0.15	0.11

(c)

n \ m	2	3	4	5
2	0.78	0.50	0.32	0.30
3	0.56	0.40	0.24	0.17
4	0.63	0.37	0.29	0.22
5	0.52	0.30	0.24	0.13

(d)

n \ m	2	3	4	5
2	0.64	0.41	0.24	0.51
3	0.55	0.45	0.38	0.40
4	0.43	0.33	0.38	0.36
5	0.44	0.41	0.26	0.25

(e)

n \ m	2	3	4	5
2	0.82	0.66	0.48	0.57
3	0.66	0.68	0.49	0.51
4	0.73	0.54	0.56	0.47
5	0.70	0.54	0.50	0.43

(f)

Table 1: Llama 2-7B and 13B generation average confidence of the correct first digit (a, b), unconditional average confidence of the correct last digit (c, d), and conditional average confidence of the correct last digit (e, f).

digit increasing when conditioned on correct prior digits, seem to be related, and could stem from the view that autoregressive language models are "exponentially diverging diffusion processes," a view that several researchers have argued informally (LeCun et al., 2023), and has also recently been more formally proven (Dziri et al., 2023). The argument is essentially that if an autoregressive LLM has some non-zero chance of making a mistake, then repeated application of that LLM to generate a long string will cause errors to compound exponentially.

This argument is not fully satisfying, however, for explaining the behavior of LLMs in predicting the last digit. Not only should $p(last\_digit|wrong\_previous\_digits)$ be the same as $p(last\_digit|correct\_previous\_digits)$ due to the computation involved (the last digit not depending on any other digits of the answer at all), but the fact that LLMs are more correct and more confident when conditioned on correct digits rather than wrong digits means that LLMs are able to internally distinguish between the two states, despite not being able to generate the entire correct string in the first place. This finding may have implications for LLMs that can backtrack and self-correct (Cundy and Ermon, 2024), as it seems that LLMs may be able to attend to subtle errors nearly immediately after generating them.

4 Conclusion

Here we present findings on the application of LLMs to arithmetic tasks, seen through the lens of Monte Carlo Dropout. We found that the abilities of what LLMs can do in practice, versus what the theory dictates should be possible for LLMs to do, can be reversed in several cases. In particular, we found that Llama 2 and Mistral could confidently and correctly output the first digit of the result of n-digit by m-digit multiplication tasks despite most likely being unable to in the general case, whereas they struggled with outputting the last digit either correctly or confidently, a task which should be easily learnable. We also found that accuracy and confidence in outputting the last digit increases when the prior digits are correct, and we believe that this finding has implications for the development of future LLM sampling or training techniques.

5 Limitations

MC Dropout is a technique that is only applicable when neural network weights are available and the neural network was trained with dropout. These restrictions limit the number of language models that can be analyzed with the techniques in this paper significantly, and crucially, state of the art language models such as GPT-4 (OpenAI, 2023), Gemini (Gemini Team et al., 2023), and Claude (Anthropic, 2023) cannot be analyzed in this way by researchers outside of OpenAI, Google, and Anthropic respectively. Such limitations make clear the need for researchers to have access to language models with open weights.

As we have restricted our analysis to Llama 2 and Mistral (which share similar architectures), it is possible that our findings do not generalize to other large language models, but given the very small number of existing language models that can be analyzed in this way, it will be difficult to gauge the generality of our findings until more language models which were trained with dropout and have open weights are released.

References

Anthropic. 2023. Model Card and Evaluations for Claude Models.

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. In *Advances in neural information processing systems 33*, pages 1877–1901.

Chris Cundy and Stefano Ermon. 2024. SequenceMatch: Imitation Learning for Autoregressive Sequence Modelling with Backtracking. In *The Twelfth International Conference on Learning Representations*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and Fate: Limits of Transformers on Compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *33rd International Conference on Machine Learning, ICML 2016*, volume 3, pages 1651–1660.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Yann LeCun, Brenden Lake, Jacob Browning, David Chalmers, Ellie Pavlick, and Gary Lupyan. 2023. Debate: Do language models need sensory grounding for meaning and understanding?

Tiedong Liu, Bryan Kian, and Hsiang Low. 2023. Goat: Fine-tuned LLaMA Outperforms GPT-4 on Arithmetic Tasks. *arXiv preprint arXiv:2305.14201*.

Shentong Mo and Miao Xin. 2023. Tree of Uncertain Thoughts Reasoning for Large Language Models. *arXiv preprint arXiv:2309.07694*.

Meredith Ringel Morris, Jascha Sohl-Dickstein, Noah Fiedel, Tris Warkentin, Allan Dafoe, Aleksandra Faust, Clement Farabet, and Shane Legg. 2024. Levels of AGI: Operationalizing Progress on the Path to AGI. *arXiv preprint arXiv:2311.02462*.

Rodrigo Nogueira, Zhiying Jiang, Jimmy Lin, and David R Cheriton. 2021. Investigating the Limitations of Transformers with Simple Arithmetic Tasks. Technical report.

OpenAI. 2023. GPT-4 Technical Report. Technical report.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Raphael Gontijo-Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. 2022.

Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. In *Advances in Neural Information Processing Systems*, volume 35.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv preprint arXiv:2302.04761*.

Artem Shelmanov, Evgenii Tsymbalov, Dmitri Puzyrev, Kirill Fedyanin, Alexander Panchenko, and Maxim Panov. 2021. How certain is your transformer? In *EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv:2305.16291*.

Noam Wies, Yoav Levine, and Amnon Shashua. 2022. Sub-Task Decomposition Enables Learning in Sequence to Sequence Tasks. In *The Eleventh International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and others. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

A Prompt Format and Hyperparameters

The exact prompt used in Sections 2.1 and 2.2 is "$111 * 472 = 52392.\ 362 * 194 = 70228.\ \{math\_question\} = \{given\_str\}$" where $math\_question$ is the multiplication task, and $given\_str$ is the empty string in Section 2.1 and all but the last digit of the correct answer in Section 2.2. In Section 2.3 the prompts are randomly generated 2-shot n-digit by m-digit multiplication examples in the same format.

We set the dropout rate to be 0.1, which is the dropout rate commonly used in GPT applications, and appears to be the dropout rate used to train Llama 2 and Mistral. All sampling from LLMs is done deterministically other than the stochasticity induced by dropout (i.e., we take argmax over logits). We collect 100 samples for each output.

B Additional experiments

B.1 Mistral-7B Ablation Experiments

We provide additional experiments with standard deviation reported on Mistral-7B in Table 2. While Mistral-7B is stronger at arithmetic tasks than both Llama 2-7B and 13B, the same patterns and conclusions found for Llama 2-7B and 13B also hold for Mistral-7B.

n \ m	2	3	4	5
2	0.97 ± 0.03	0.98 ± 0.03	0.98 ± 0.02	1.00 ± 0.00
3	0.98 ± 0.03	1.00 ± 0.00	0.94 ± 0.09	0.93 ± 0.04
4	0.99 ± 0.01	0.87 ± 0.15	0.98 ± 0.04	0.82 ± 0.09
5	0.89 ± 0.1	0.94 ± 0.11	0.95 ± 0.06	0.99 ± 0.01

(a)

n \ m	2	3	4	5
2	0.74 ± 0.06	0.57 ± 0.26	0.52 ± 0.29	0.41 ± 0.21
3	0.87 ± 0.10	0.70 ± 0.13	0.20 ± 0.12	0.11 ± 0.07
4	0.44 ± 0.14	0.70 ± 0.14	0.28 ± 0.23	0.30 ± 0.15
5	0.70 ± 0.10	0.33 ± 0.09	0.20 ± 0.13	0.22 ± 0.07

(b)

n \ m	2	3	4	5
2	0.85 ± 0.23	0.83 ± 0.13	0.73 ± 0.21	0.76 ± 0.23
3	0.86 ± 0.13	0.85 ± 0.11	0.75 ± 0.22	0.57 ± 0.32
4	0.76 ± 0.17	0.62 ± 0.27	0.77 ± 0.26	0.59 ± 0.26
5	0.80 ± 0.18	0.68 ± 0.21	0.65 ± 0.26	0.55 ± 0.35

(c)

Table 2: Mistral-7B generation average and standard deviation confidence of the correct first digit (a), unconditional average and standard deviation confidence of the correct last digit (b), and conditional average and standard deviation confidence of the correct last digit (c).