What is a Number, That a Large Language Model May Know It?

Raja Marjieh[1,†,*] **Veniamin Veselovsky**[2,†], **Thomas L. Griffiths**[1,2,‡], **Ilia Sucholutsky**[3,‡]
[1]Department of Psychology, Princeton University
[2]Department of Computer Science, Princeton University
[3]Center for Data Science, New York University

Abstract

Numbers are a basic part of how humans represent and describe the world around them. As a consequence, learning effective representations of numbers is critical for the success of large language models as they become more integrated into everyday decisions. However, these models face a challenge: depending on context, the same sequence of digit tokens, e.g., 911, can be treated as a number or as a string. What kind of representations arise from this duality, and what are its downstream implications? Using a similarity-based prompting technique from cognitive science, we show that LLMs learn representational spaces that blend string-like and numerical representations. In particular, we show that elicited similarity judgments from these models over integer pairs can be captured by a combination of Levenshtein edit distance and numerical Log-Linear distance, suggesting an entangled representation. In a series of experiments, we show how this entanglement is reflected in the latent embeddings, how it can be reduced but not entirely eliminated by context, and how it can propagate into a realistic decision scenario. These results shed light on a representational tension in transformer models that must learn what a number is from text input.

1 Introduction

"What is a number, that a man may know it, and a man, that he may know a number?" *Warren McCulloch* (1961)

Numbers play a pivotal role in many aspects of human cognition [2]. Starting from a narrow sense of magnitude in infancy, humans gradually acquire nuanced representations of number that capture abstract properties such as "even" and "odd" [3, 4]. Decades of research have been devoted to understanding how humans process and represent numbers [3, 5–8]. In fact, this endeavor dates back to some of the earliest applications of neural networks to artificial intelligence [1, 9] when researchers sought to define computing machines (a simulated "man") that can intelligently process numbers. The question of how computing machines represent number assumes key importance as modern neural networks, in particular large language models (LLMs), are becoming integrated into everyday decisions [10–15]. Since these models learn representations by predicting tokens in text, LLMs face a challenge: depending on context, the same sequence of digit tokens, e.g., 101 or 911, can be treated as a number or as a string. This duality introduces an issue akin to polysemy and homonymy [16], but it extends across different symbolic systems. What kind of representations arise from this duality, and what are its downstream implications? This is not straightforward to answer, in part because LLMs lack the experience that shapes humans' number representations, and in part because model internals are not always available and diagnostic tests can be challenging to design.

[*]Corresponding author. Email: raja.marjieh@princeton.edu.
[†/‡]Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: CogInterp 2025.

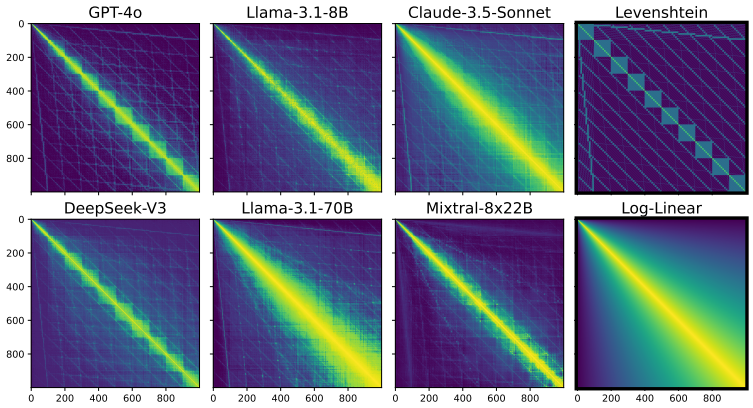

Figure 1: LLM number similarity matrices (symmetrized) over all integer pairs in the range $0 - 999$, along with two theoretical similarity matrices derived from a Levenshtein string edit distance and a psychological Log-Linear numerical distance (highlighted in black).

Here we address this challenge by leveraging tools from cognitive science for characterizing representations [17–21]. Specifically, we elicit similarity judgments across number pairs from six modern LLMs using an appropriate prompt. We then use those judgments to construct detailed maps (similarity matrices) that capture how the models organize numbers. Crucially, this technique can be applied to any model without the need to access its internals. Moreover, when the model internals are available, this approach can be combined with probing techniques to evaluate how prompt behavior is reflected in the structure of the latent embeddings (for a survey of related work see Appendix A).

We show that despite variation in model training data and size, all models exhibit highly regular patterns that can be adequately decomposed in terms of a psychologically-motivated numerical distance based on a Log-Linear representation of numbers [8] and a string edit distance, suggesting an entanglement. We show how this entanglement can be reduced but not eliminated by introducing a context that specifies the 'type' of the integer (`int()` vs. `str()`), and how it can be probed internally in a model's latent space. Inspired by these results, we then construct a decision scenario that reveals how string-bias can propagate into a realistic setting leading to incorrect answers. Viewed together, these findings shed light on an intrinsic tension between number and string representations that modern large language models must learn to navigate.

2 Experiments

Our approach builds on the paradigm of similarity judgments from cognitive science [17, 19–22], which is also closely related to representational similarity analysis (RSA) from neuroscience [23]. Given a domain of interest $\mathcal{D}$, a set of representative items from that domain $\{x_1, x_2, \ldots, x_N\} \subset \mathcal{D}$ (or 'stimuli'), and an agent $\mathcal{A}$ whose representation $\mathcal{M}(\mathcal{D})$ one would like to characterize, the paradigm proceeds by eliciting pairwise similarity judgments from $\mathcal{A}$ across all pairs of items ('How similar is the item x_i to the item x_j?'). These judgments are then aggregated into a similarity matrix s_{ij} which defines a notion of proximity on $\mathcal{D}$ that can be used to characterize $\mathcal{M}(\mathcal{D})$. Since similarity is by construction neutral, it leaves it to the agent to impose its own structure on s_{ij}.

Our domain of interest is the set of non-negative integers $\{0, 1, 2, \ldots\}$, and the agent is an LLM. We consider a representative sample of widely-used models, namely, GPT-4o [24], two variants of Llama-3.1 (8b and 70b) [25], DeepSeek-V3 [26], Claude-3.5-Sonnet [27], and Mixtral-8x22B [28]. We then apply the similarity technique in a series of experiments that span different contexts as well as behavioral (prompt-level) and internal (embedding-level) analyses and examine how the observed patterns decompose in terms of theoretical string and numerical metrics. We detail those experiments in what follows and lay out the full methodology in Appendix B.

2.1 Eliciting Similarity Judgments

In the first experiment, we elicited similarity judgments across all pairs of integers in the range $0 - 999$ ('How similar are the two numbers?'; see Appendix D for the full prompt). We begin by presenting the qualitative patterns in the raw data and then quantify them using a suitable regression analysis. Figure 1 shows the similarity matrices for all models (see Methodology). The resulting

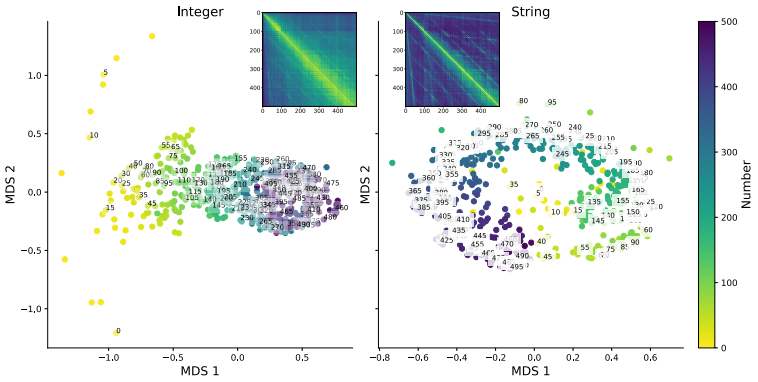

Figure 2: Decoded string and integer subspaces from Llama-3.1-8b using linear probes (see Methodology). The decoded similarity matrices are provided as insets along with their multidimensional scaling solutions (MDS). Integers are labeled every 5 points.

LLM matrices are highly structured with a dominant area around the diagonal that in some cases (e.g., GPT-4o and DeepSeek-V3) takes a block-diagonal form, in addition to other sub-diagonal structures. These patterns appear to blend the properties found in two theoretical similarity matrices associated with string edit distance (Levenshtein) and numerical distance (Log-Linear) highlighted in the rightmost panels of Figure 1 (see Methodology). To further see whether these patterns arise from a tension between string and integer representations, in a following experiment we resolved the ambiguity of the similarity prompt by specifying the 'type' of the token using `int()` and `str()` contexts (see Appendix D for prompts). The results are shown in Figure S3A. In this case, we found that the context intervention pushed the matrices in opposing directions. For example, GPT-4o loses its block diagonal structure in the `int()` context, whereas Claude-3.5-Sonnet gains it in the `str()` context. Similar trends can also be found in the other models.

To quantify the above, we regressed the Levenshtein and Log-Linear distance measures against the LLM similarity judgments in the three contexts considered. A breakdown of the explained variance (coefficient of determination R^2) per condition are provided in Figure S3B. Overall, in the default context we found that a linear combination of the numerical and Levenshtein distances is sufficient to achieve an average R^2 of .726 (95% CI of mean: $[.725, .726]$), with the separate components each contributing significantly (Log-Linear only R^2 CI: $[.607, .609]$, Levenshtein only R^2 CI: $[.213, .215]$). We note that replacing the Log-Linear distance with a simple linear ℓ_1 distance diminishes combined average performance to an R^2 of .567 (95% CI: $[.567, .568]$; see Figure S5). In the other contexts, the metrics were correspondingly pushed in opposing directions. In the `str()` case, the mean R^2 CIs were: Combined: $[.620, .621]$, Log-Linear: $[.410, .412]$, and Levenshtein: $[.309, .311]$. On the other hand, in the `int()` context these were: Combined: $[.721, .722]$, Log-Linear: $[.645, .646]$, and Levenshtein: $[.156, .158]$ (see Appendix I.1 for experiments with other number bases).

2.2 Probing Internal Representations

The previous experiments showed that when an LLM is prompted for similarity judgments between two different integer tokens, the resulting behavioral metric is a combination of string and integer distance. In this section, we show how this also holds on the representational (i.e., token-embedding) level in a model for which we have internal access (Llama-3.1-8b). To do that, we train linear probes from the last token in the prompt to decode the Log-Linear and Levenshtein distances between the integers in question (see Methodology). This is non-trivial as there is no guarantee that such a linear transformation exists unless the model encodes the relevant information in its embedding (this can be seen also from the fact that our probes had 4,096 parameters, corresponding to the latent dimension, fitted on 9,500 data points, and then evaluated on 250,000 data points; see Methodology). In Figure 2 we illustrate the decoded similarity matrices from the embeddings. Here we notice a pattern that is consistent with the behavioral (prompt-based) data. In the string probe case (right panel in Figure 2), the model is able to capture the edit-distance similarity between two numbers; whereas in the integer probe case (left panel in Figure 2), the decoded pattern is indeed much more log-linear as can be seen from the diagonal area in the matrix. Interestingly, however, string similarities still bleed into the representation as can be seen from the sub-diagonal structures. Quantitatively, the Pearson correlation between the string probe and the Levenshtein and Log-Linear measures is .650 and .527,

respectively. For the integer probe, the correlation is .917 with the Log-Linear distance and .393 with the Levenshtein distance (see Table 2 in the Appendix). We also apply multidimensional scaling [17] to the decoded similarity matrices to derive two-dimensional maps of the relations between integers (Figure 2). We see that the probes nicely dissociate the integer and string subspaces. In the integer subspace, numbers are organized on a scale that captures numerical distance. In the string subspace, the scale is replaced with a non-linear pattern that closely follows edit distance.

2.3 Behavioral Implications

In the final experiment, we wanted to see whether there are behavioral implications for the number-string tension in a naturalistic scenario. To do that, we presented the models with an empirical setting in which they required a test tube with a certain unavailable compound concentration (given in parts per million units, ppm; full prompt is provided in Appendix D). The models were asked to choose from two available test tubes the one containing the most similar concentration to the one desired. Crucially, we constructed diagnostic concentration triplets that lead to divergent answers depending on whether a Levenshtein distance is used or a Log-Linear one (see Methodology). Here is an example prompt: 'You require a compound with a concentration of approximately 785 ppm. Two test tubes are available: one containing 685 ppm and the other 791 ppm. Your task is to determine which test tube provides the most similar concentration to your required dosage.' Clearly, a concentration of 791 ppm is much more similar to 785 ppm than 685 ppm, but a model with strong string bias may erroneously choose 685 ppm which shares more digits with 785 ppm. We then evaluated the percentage of incorrect (Levenshtein-consistent) answers as a measure of string bias. In addition to the 3-digit triplets, we also considered 5-digit ones to see whether the problem may be exacerbated by the inclusion of more digits (e.g., 22565, 12565, and 28743; see Methodology). We also considered both orders of presentation to see whether there are any ordering effects when the Levenshtein-aligned option is presented first vs. the Log-Linear one ('Reverse' setting).

We found that all models had some degree of string-bias errors but the range varied greatly and was enhanced for longer numbers (see Figure S4). In the 3-digit case, the largest percentage (averaged over order) was 36.86% for Llama-3.1-8b (followed by 11.38% for Llama-3.1-70b), and the smallest was 0.02% for GPT-4o. In the 5-digit case, the largest percentage was 47.03% for Llama-3.1-8b (followed by 19.04% for Llama-3.1-70b), and the smallest was 0.41% for GPT-4o. Interestingly, the Llama-3.1 models exhibited larger degrees of bias, as well as some degree of order effects in some cases (note that this cannot be attributed to a generic presentation bias since in that case we would not expect to see a difference between the 3- and 5-digit scenarios). Overall, these results suggest that the number-string tension can manifest itself in a realistic context, and that the extent to which context can suppress the stringiness behavior varies across models.

3 Discussion

We have provided evidence for a representational tension in large language models that learn to represent numbers by processing text input. This tension is reflected externally (through prompt-elicited similarity judgments), internally (through embedding probes), and could also propagate into a quantitative decision scenario. Our results suggest that the tension may arise from ambiguity in the 'type' of digit tokens in a given context. This ambiguity is reminiscent of polysemy and homonymy in language [16] whereby a single word can have multiple meanings, but in our case the ambiguity is extended across different systems. This entanglement of types is also reflected in our probing results whereby the linearly decoded similarity matrices had some residual shared structure.

Our findings are consistent with recent probing work showing that LLMs are able to encode the value of numbers [10]. Interestingly, however, we found that when the setting is not strictly mathematical as in arithmetic, LLMs appear to exhibit a more psychologically plausible log-linear representation, though blended with string distance. This is consistent with recent work showing that LLMs encode rich perceptual knowledge [29]. The results also make contact with the literature on how LLMs perform numerical operations [11, 12] and the possible sources of numerical errors that could arise from their mode of learning [13, 14]. One possibility is that LLMs attempt to categorize the type of an integer token from context, and then use that to specify a set of operations on units of that type. For example, if an LLM views the numbers 1 and 2 as strings, then asking it to sum the two numbers may result in 12 rather than 3. Future work could interrogate whether there is indeed a causal link between such type-categorization mechanisms and numerical errors.

We end by discussing some limitations that point towards future directions. First, due to resource limitations our LLM results were restricted to zero temperature and the probing analysis was limited

to the smaller Llama-3.1-8b model. Future work could look into how the results generalize to a stochastic sampling setting and other models for probing. Second, while we explored the effect of context on number similarity judgments, it remains to be seen whether one could causally intervene at the embedding level to steer the model from one token type to another. Third, while our naturalistic decision scenario was designed to be as diagnostic as possible, further work is necessary to assess how pervasive this issue is in more generic daily numerical use cases. Relatedly, reasoning-based models are increasingly becoming the *de facto* standard in user-facing applications (e.g., OpenAI's o1 [30], DeepSeek's R1 [31]). Extending our findings to these models is a promising avenue for future work, as their ability to reason may fundamentally alter how similarity judgments are produced. Finally, our work was restricted to the domain of numbers but our results may be a special case of a more broad symbol-string duality that impacts other domains (e.g., code and special characters). We hope to report on these directions in the future.

As artificial intelligence becomes increasingly intermingled with human intelligence, knowing what a number is becomes an increasingly important concern. McCulloch [1] argued that the human capacity to understand numbers could be acquired by machines based on artificial neural networks. Our results suggest that there are still meaningful differences in how humans and large language models based on artificial neural networks conceive of numbers.

Acknowledgments. This work was supported by an Azure Foundation Models Research grant from Microsoft.

References

[1] Warren S McCulloch. What is a number, that a man may know it, and a man, that he may know a number. *General Semantics Bulletin*, 26(27):7–18, 1961.

[2] Stanislas Dehaene. *The number sense: How the mind creates mathematics*. Oxford University Press, 2011.

[3] Kevin Miller and Rochel Gelman. The child's representation of number: A multidimensional scaling analysis. *Child development*, pages 1470–1479, 1983.

[4] Steven T Piantadosi, Julian Jara-Ettinger, and Edward Gibson. Children's learning of number words in an indigenous farming-foraging group. *Developmental science*, 17(4):553–563, 2014.

[5] Andreas Nieder and Stanislas Dehaene. Representation of number in the brain. *Annual review of neuroscience*, 32(1):185–208, 2009.

[6] Samuel J Cheyette and Steven T Piantadosi. A unified account of numerosity perception. *Nature human behaviour*, 4(12):1265–1272, 2020.

[7] Joshua Tenenbaum. Rules and similarity in concept learning. *Advances in neural information processing systems*, 12, 1999.

[8] Steven T Piantadosi. A rational analysis of the approximate number system. *Psychonomic bulletin & review*, 23:877–886, 2016.

[9] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[10] Fangwei Zhu, Damai Dai, and Zhifang Sui. Language models encode the value of numbers linearly. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 693–709, 2025.

[11] Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[12] Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. *arXiv preprint arXiv:2305.15054*, 2023.

[13] R Thomas McCoy, Shunyu Yao, Dan Friedman, Mathew D Hardy, and Thomas L Griffiths. Embers of autoregression show how large language models are shaped by the problem they are trained to solve. *Proceedings of the National Academy of Sciences*, 121(41):e2322420121, 2024.

[14] R. Thomas McCoy, Shunyu Yao, Dan Friedman, Mathew D. Hardy, and Thomas L. Griffiths. When a language model is optimized for reasoning, does it still show embers of autoregression? An analysis of OpenAI o1. *arXiv preprint arXiv:2410.01792*, 2024.

[15] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *arXiv preprint arXiv:2305.14975*, 2023.

[16] Agustín Vicente and Ingrid L Falkum. Polysemy. In *Oxford research encyclopedia of linguistics*. 2017.

[17] Roger N Shepard. Multidimensional scaling, tree-fitting, and clustering. *Science*, 210(4468): 390–398, 1980.

[18] Roger N Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987.

[19] Amos Tversky and J Hutchinson. Nearest neighbor analysis of psychological spaces. *Psychological review*, 93(1):3, 1986.

[20] Joshua B Tenenbaum and Thomas L Griffiths. Generalization, similarity, and Bayesian inference. *Behavioral and brain sciences*, 24(4):629–640, 2001.

[21] Raja Marjieh, Nori Jacoby, Joshua C Peterson, and Thomas L Griffiths. The universal law of generalization holds for naturalistic stimuli. *Journal of Experimental Psychology: General*, 153 (3):573, 2024.

[22] Roger N Shepard. The analysis of proximities: multidimensional scaling with an unknown distance function. i. *Psychometrika*, 27(2):125–140, 1962.

[23] Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:249, 2008.

[24] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[25] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[26] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[27] AI Anthropic. Claude 3.5 Sonnet model card addendum. *Claude-3.5 Model Card*, 3:6, 2024.

[28] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[29] Raja Marjieh, Ilia Sucholutsky, Pol van Rijn, Nori Jacoby, and Thomas L Griffiths. Large language models predict human sensory judgments across six modalities. *Scientific Reports*, 14 (1):21445, 2024.

[30] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[31] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

[32] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[34] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*, 2019.

[35] Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054, 2020.

[36] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8: 842–866, 2021.

[37] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

[38] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[39] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In Yonatan Belinkov, Sophie Hao, Jaap Jumelet, Najoung Kim, Arya McCarthy, and Hosein Mohebbi, editors, *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 16–30, 2023. doi: 10.18653/v1/2023.blackboxnlp-1.2.

[40] Wes Gurnee and Max Tegmark. Language models represent space and time. *arXiv preprint arXiv:2310.02207*, 2024.

[41] Kai Nylund, Suchin Gururangan, and Noah A Smith. Time is encoded in the weights of finetuned language models. *arXiv preprint arXiv:2312.13401*, 2023.

[42] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

[43] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

[44] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

[45] Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D. Goodman. Finding alignments between interpretable causal variables and distributed neural representations. *arXiv preprint arXiv:2303.02536*, 2024.

[46] Marcel Binz and Eric Schulz. Using cognitive psychology to understand GPT-3. *Proceedings of the National Academy of Sciences*, 120(6):e2218523120, 2023.

[47] Xuechunzi Bai, Angelina Wang, Ilia Sucholutsky, and Thomas L Griffiths. Explicitly unbiased large language models still form biased associations. *Proceedings of the National Academy of Sciences*, 122(8):e2416228122, 2025.

[48] Taylor Webb, Keith J Holyoak, and Hongjing Lu. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541, 2023.

[49] VI Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Proceedings of the Soviet physics doklady*, 1966.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

A Related Work

A.1 Numbers and Language Models

The processing and representation of numbers in large language models is a topic of active research. Number representations have recently been studied within pretrained language models using techniques from mechanistic interpretability. For example, Zhu et al. [10] devised a dataset of addition problems to show that LLMs encode the value of numbers linearly in that context. Likewise, circuit analysis has been adopted to characterize how smaller language models compute "greater than" [11] and basic arithmetic operations [12]. Other work studied how simple models solve modular arithmetic tasks [32]. More behavioral approaches have also been explored. Tian et al. [15] looked at how model generations of numbers can be used to test for answer calibration. Likewise, McCoy et al. (2024, 2024) showed how a Bayesian framework can be used to characterize how LLM performance on various tasks, including numeric ones, depends on the probability of the input, task, and output.

A.2 Probing and Representation Analysis in LLMs

Probing is a well-established technique for studying the internal representations of language models. Dating back to BERT [33], probing has been used to explore various aspects of representation such as how models encode sentence structure [34–36]. More recently, probing has become a common technique in studying larger language models. In the work by Zhu et al. [10] mentioned earlier, the authors designed linear probes to extract number value from latent embeddings. Probes have also been used to discover how language models process concepts across layers [37], and for extracting uncertainty in language model generations [38]. Similarly, probes, and latent analysis more broadly, have been used in the world models literature, e.g., to show that language models trained on Othello games develop a world model of the game [39], and that they learn to represent space and time [40, 37, 41]. More recent approaches to studying representations in language models have also been proposed like activation patching [42], sparse autoencoders [43, 44], and distributed alignment search [45].

A.3 Behavioral Analysis of Language Models

By combining the prompt comprehension capabilities of large language models with the wide range of paradigms available in the behavioral sciences, a growing line of work proposes new tools for characterizing behavior in LLMs. For example, Marjieh et al. [29] used a suite of perceptual judgment tasks from psychophysics to characterize sensory knowledge in LLMs. Binz and Schulz [46] leveraged paradigms from cognitive psychology to study how GPT-3 solves various tasks. Bai et al. [47] used tools from social psychology to diagnose implicit racial and gender bias in LLMs. Webb et al. [48] subjected LLMs to analogical tasks to study their abstract reasoning capacities. Our work takes a similar approach to this line of work, and combines it with modern probing techniques to study LLM behavior and representation in the domain of number.

B Methodology

B.1 Tasks

Our experiments fall into three categories, all of which are associated with a certain behavioral prompt for eliciting LLM judgments over numerical quantities. The prompts are provided in Appendix D. In the first category, we elicited similarity judgments over all pairs of numbers in the range $0 - 999$ ('How similar are the two numbers on a scale of 0 (completely dissimilar) to 1 (completely similar)?'; see Appendix D). We then constructed symmetric similarity matrices s_{ij} by averaging over both presentation orders (i.e., $s(x_i, x_j)$ and $s(x_j, x_i)$), and then evaluated the structure that emerged (see Evaluation Metrics below) and the way it is affected by context. We focused on the range $0 - 999$ because all integers within it are represented as unique tokens in the models considered, which controls for any tokenization-specific differences. In the second category, we extended the similarity analysis for one of the models for which we had internal access (Llama-3.1-8b) and probed the extent to which its internal token embeddings reflected the external behavior through a linear transformation (see Probing Language Models). Finally, in the third category we constructed a naturalistic decision scenario in which the model had to select from two available numerical quantities q_1, q_2 (a compound

concentration in a test tube) the one that is most similar to a desired quantity q_0 (further details in Experiments and Appendix D). We constructed such triplets (q_0, q_1, q_2) in a way that the answer would diverge depending on whether the quantities are treated as numbers or strings (see Constructing Close Triplets below) which then allowed us to probe the string-bias of models.

B.2 Models

We evaluated four open-source and two closed-source models. The open-source models consist of Llama-3.1-8b-Instruct, Llama-3.1-70b-Instruct, DeepSeek-V3, Mixtral-8x22B-Instruct-V0.1, and the closed source include GPT-4o and Claude-3.5-Sonnet. Henceforth, we will refer to the Llama and Mixtral models as Llama-3.1-8b, Llama-3.1-70b, and Mixtral-8x22B to improve readability. Since generating 1000×1000 similarities requires sampling 1,000,000 pairwise comparisons which costs over $160 for Claude-3.5-Sonnet, we run the models once at a temperature of zero to get the most likely answer across the different settings.

B.3 Evaluation Metrics

To characterize the extent to which the derived similarity matrices s_{ij} were string-like or number-like, we regressed their values against two theoretical distance measures, namely, the Levenshtein string edit-distance $d_{Lev}(a, b)$ [49] and the psychological Log-Linear number distance $d_{Log}(a, b)$ [8] (we also considered a simple linear ℓ_1 distance $|a - b|$ but found that it led to worse results; Figure S5). The Log-Linear distance captures the psychological phenomenon whereby humans represent larger numbers with less fidelity leading to a logarithmic sensitivity to magnitude, i.e. $x - y \to \log(x) - \log(y)$ (see Appendix E for definition). The Levenshtein distance, on the other hand, is defined as the minimum number of deletions, insertions, or substitutions required to transform one string to the other. This can be defined recursively (see Appendix E) and we used the Python package `Levenshtein`[2] to compute it. For example, the Levenshtein distance between the digits 200 and 100 is 1 because it takes a single substitution ($2 \to 1$) to transform one string into the other.

Given the two distance measures evaluated on the range of interest $d_{ij}^{(Lev)}$ and $d_{ij}^{(Log)}$, we transformed them into similarity scores $s_{ij}(d) = 1 - d_{ij}/\max\{d_{ij}\}$ and then fitted them to s_{ij} (after z-scoring), i.e., $s_{ij} = \alpha + \beta s_{ij}^{(Lev)} + \gamma s_{ij}^{(Log)}$ using the LinearRegression method from `scikit-learn` [50] and computed the coefficient of determination (R^2 or variance-explained) which we also repeated for the individual metrics separately. To derive confidence intervals, we bootstrapped over the number pairs (since the models had zero temperature) with replacement and 1,000 repetitions. All analysis code is provided in the accompanying repository (see Reproducibility).

B.4 Probing Language Models

When generating an output, a language model first tokenizes the input sentence into a sequence of tokens $x_1, ..., x_n \in V$ where V is some vocabulary. Then during the forward pass, the model incrementally transforms the initial embeddings (extracted from a vocabulary embedding layer) through each layer. The internal representations at each layer — commonly referred to as *residuals* — capture evolving latent features. These residuals can be analyzed to probe specific properties of the input or model behavior.

In our case, we train an affine transformation for each layer within the model that takes in a specific latent $h_i^{(j)}$ for token i at layer j, and then learns a vector w and bias term b, such that $w \cdot h_i^{(j)} + b$ predicts a specific distance measure. Due to computational constraints, the probes are run only on the Llama-3.1-8b model. We take the original similarity prompt and extract the residuals from the last token — in our case the ":" after "Result:" (see Appendix D) — and train two linear probes on them. One probe is trained to predict the Levenshtein distance between the two inputted numbers, the other the Log-Linear distance.

We train the probes on 9,500 random pairs of numbers between 0 and 999, and evaluate on the range 0-500. We ablate across all layers and select layer 25 for the final probe (see Appendix for ablations on number of training points and performance across layers). We then run these probes on all 500×500 (250,000) combinations of number similarities and report their corresponding correlations.

[2] `https://rapidfuzz.github.io/Levenshtein/`

To illustrate the results of the probe, we apply multidimensional scaling (MDS) [22] on the 500×500 similarity judgments extracted by the probe. MDS is a visualization technique that represents high-dimensional data in a lower-dimensional space, preserving the relative pairwise distances or dissimilarities as closely as possible. We computed MDS embeddings using `scikit-learn`.

B.5 Constructing Close Triplets

Our goal was to construct diagnostic triplets for the naturalistic decision scenario such that (i) the options are numerically close to each other but there is an unambiguous correct answer, and (ii) the options are very different in terms of their edit distance relative to the target. To do that, we constructed the target quantity q_0 by randomly sampling three digits from the set $\{2, \ldots, 9\}$ with replacement and combining them into a number (e.g., $3, 3, 1 \rightarrow 331$). Then, we constructed a Levenshtein-aligned option by subtracting 1 from the largest decimal entry (i.e., $331 \rightarrow 231$). Finally, we constructed a Log-aligned option by keeping the largest decimal entry from the target and randomly sampling two new integers from the range $\{1, \ldots, 9\}$ excluding the digits that appeared in the other two numbers (e.g., 357). Thus, the result would be $(331, 231, 357)$ in our example which satisfies the desired properties. We repeated the process also for five digit numbers to probe how things change for longer numbers (e.g., $25337, 15337, 26886$). We sampled 10,000 such triplets from each length category and took the unique subset. Overall, there were 6,474 unique three digit triplets, and 9,995 five digit triplets.

Reproducibility. Code for reproducing the analyses can be accessed here: `https://github.com/vminvsky/numbers-in-llms`

C Supplementary Figures

We resolved the ambiguity of the similarity prompt by specifying the 'type' of the token using `int()` and `str()` contexts (see Appendix D for prompts). The results are shown in Figure S3A. We regressed the Levenshtein and Log-Linear distance measures against the LLM similarity judgments in the three contexts considered. A breakdown of the explained variance (coefficient of determination R^2) per condition are provided in Figure S3B.

D Prompts

Basic Similarity Prompt

How similar are the two numbers on a scale of 0 (completely dissimilar) to 1 (completely similar)? Respond only with the rating.
Number: {NUM1}
Number: {NUM2}
Rating:

Integer Similarity Prompt

How similar are the two numbers on a scale of 0 (completely dissimilar) to 1 (completely similar)? Respond only with the rating.
Number: **int**({NUM1})
Number: **int**({NUM2})
Rating:

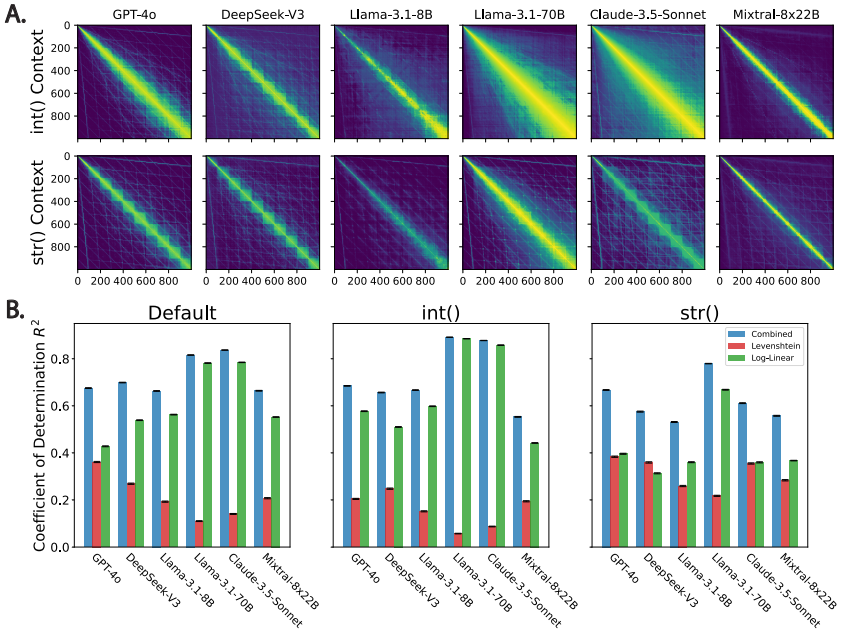

Figure S3: Context effects on LLM-elicited number similarity matrices and their decomposition. **A**. LLM similarity matrices under the effect of 'type' specification: `int()` vs. `str()` (see Appendix D for prompts). **B**. Coefficient of determination (R^2) for the different similarity matrices under the default (Figure 1), `int()`, and `str()` contexts for the combined and separate Levenshtein (string) and Log-Linear (numerical) distance predictors (error bars indicate 95% CIs; see Methodology).

String Similarity Prompt

How similar are the two numbers on a scale of 0 (completely dissimilar) to 1 (completely similar)? Respond only with the rating.
Number: **str**({NUM1})
Number: **str**({NUM2})
Rating:

Different Base Similarity Prompt

How similar are the two numbers on a scale of 0 (completely dissimilar) to 1 (completely similar)? Respond only with the rating.
Base {BASE} number: {NUM1}
Base {BASE} number: {NUM2}
Rating:

Compound Concentration Prompt

You require a compound with a concentration of approximately {NUM1} ppm. Two test tubes are available: one containing {NUM2} ppm and the other {NUM3} ppm. Your task is to determine which test tube provides the most similar concentration to your required dosage. Which one will you choose? Respond only with the ppm value of the test tube you choose.

E Theoretical Distance Measures

We defined the Log-Linear psychological distance measure between two numbers x and y using the formula

$$d_{Log}(x, y) = 1 - \exp\left(-|\log(x + \epsilon) - \log(y + \epsilon)|\right)$$

where $\epsilon = 10^{-4}$ is a small regularizer to account for the fact that our domain included 0.

The Levenshtein edit distance $d_{Lev}(a, b)$ between two strings of characters $a = a_0 \dots a_n$ and $b = b_0 \dots b_n$ is defined recursively as

$$d_{Lev}(a,b) = \begin{cases} |a|, & \text{if} \quad |b| = 0 \\ |b|, & \text{if} \quad |a| = 0 \\ d_{Lev}(a_{1\dots n}, b_{1\dots n}), & \text{if} \quad a_0 = b_0 \\ 1 + \min\left\{d_{Lev}(a_{1\dots n}, b), d_{Lev}(a, b_{1\dots n}), d_{Lev}(a_{1\dots n}, b_{1\dots n})\right\}, & \text{else} \end{cases}$$

F Model versions

For reproducibility, we include the model versions used to collect data in Table 1, alongside the providers.

Model Name	Model Version	Provider
Claude-3.5-Sonnet	claude-3-5-sonnet-20241022	Claude API
DeepSeek-V3	DeepSeek-V3	together.ai
GPT-4o	gpt-4o-2024-08-06	Azure API
Llama-3.1-8b	Llama-3.1-8b-Instruct	together.ai
Llama-3.1-70b	Llama-3.1-70b-Instruct	together.ai
Mixtral-8x22B	Mixtral-8x22B-Instruct-V0.1	together.ai

Table 1: Full model versions alongside the model provider used to access them.

G Probing Analysis

G.1 Background on Language Models

When a sentence s is fed into a language model, the text is first tokenized into a sequence of tokens, $x_1, ..., x_t \in V$, where V is the vocabulary of the model. Then when generating the x_{t+1} token, the model, $f : \mathcal{X} \to \mathcal{Y}$ maps the input sequence to a probability distribution $\mathcal{Y} \in \mathbb{R}^{|V|}$. This is done by first embedding each of the input tokens x_i using a learned input embedding matrix E that maps the vocabulary to a set of embeddings denoted by $h_i^{(0)}$. As the token is processed throughout the model, each token's latent layer is updated as follows:

$$h_i^{(j)} = h_i^{(j-1)} + g(h_1^{(j-1)}, ..., h_i^{(j-1)})$$

Here g is a function that typically consists of an MLP and an attention layer, alongside some form of normalization. Using the final representation of the last token $h_x^{(L)}$, an unembedding matrix is applied, W, which converts the latent vector back into the vocabulary space. We use the internal latents of the language model to train the probe.

G.2 Training and Evaluation

The input data to the probe is the residuals of the last token from the different layers. In our case, this maps to the ":" after "Rating". It is important to note that instruct models require specific prompts using roles to stay faithful to the training process. To account for this, we put the original task into the user message, and put the "Rating:" as the first tokens of the assistant message. We then continued generating from these forced tokens.

	String Probe	Levenshtein	Int Probe	Log-Linear
String Probe	1	–	–	–
Levenshtein	0.650^{***}	1	–	–
Int Probe	0.667^{***}	0.393^{***}	1	–
Log-Linear	0.527^{***}	0.266^{***}	0.917^{***}	1

Table 2: Correlation between the probes and the groundtruth number similarities based on Levenshtein and Log-Linear distance. All correlations have $p < 0.00001$.

The probe itself consisted of a single affine layer predicting one value (the similarity score). In other words, by learning the probe we were essentially learning a mapping from the hidden dimension of the model (4096) to a one-dimensional space.

To train the probe, we first extract residuals from 10,000 random pairs of numbers between 0-999 across all 33 layers of Llama-3.1-8b. We then train probes on all layers and with varying dataset sizes (see below for the analysis). The groundtruth data is calculated by measuring either the Log-Linear or Levenshtein distance between all pairs of numbers. Then the probe is trained to predict these scores. The probe itself is trained with Adam for 100 epochs.

In Table 2 we report the correlation between the probes and the groundtruth Levenshtein and log-linear judgments.

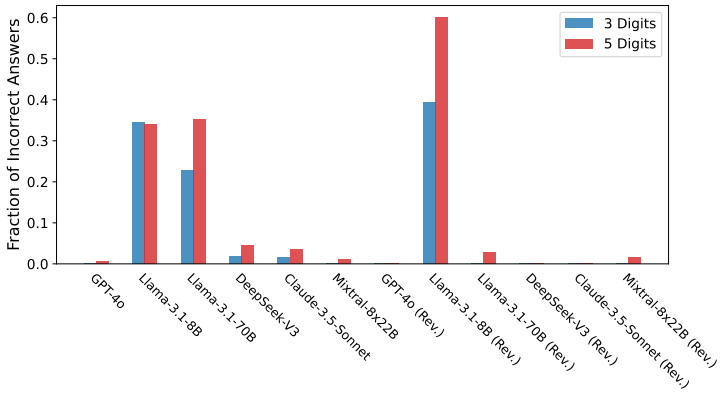

Figure S4: Probing string-bias in a naturalistic decision scenario. Bar plots indicate the faction of times an incorrect (Levenshtein-aligned) option was chosen for the 3-digit and 5-digit scenarios considered, and the two possible presentation orders (see Methodology). '(Rev.)' indicates the case in which the Levenshtein-aligned option was presented second (see prompt in Appendix D).

H Linear Number Distance Control Analysis

As an additional control we reran our regression analysis with a linear ℓ_1 number distance $|a - b|$ rather than the psychologically-motivated log linear one. The results are shown in Figure S5 and they yielded a worse fit relative to the Log-Linear case.

I Extensions

I.1 Other Bases

We evaluated the extent to which the string-like similarity generalizes to less common number bases. The idea here is that if the model is relying on edit distance, then the effect should persist irrespective of how uncommon the underlying numerical basis is. To that end, we elicited additional similarity judgments for a subset of three models (GPT-4o, Llama-3.1-8b, and Llama-3.1-70b) using two additional number bases: base 4 and base 8 (see Methodology). The results are shown in

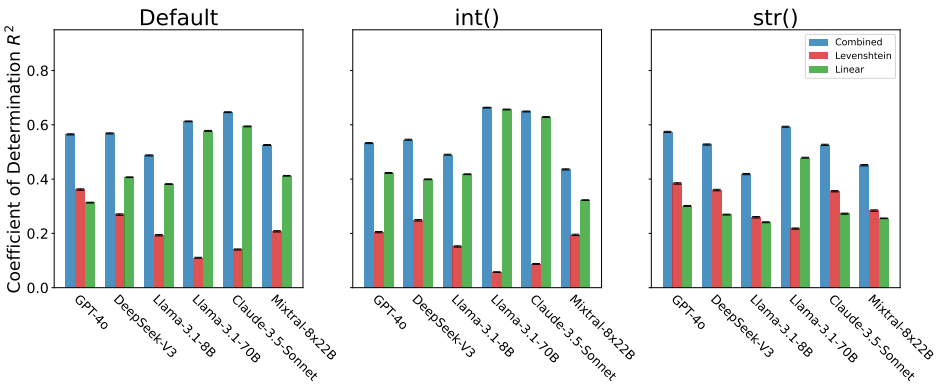

Figure S5: Coefficient of determination (R^2) for the different similarity matrices under the default (Figure 1), `int()`, and `str()` contexts for the combined and separate Levenshtein (string) and Linear (numerical) distance predictors (error bars indicate 95% confidence intervals; see Methodology).

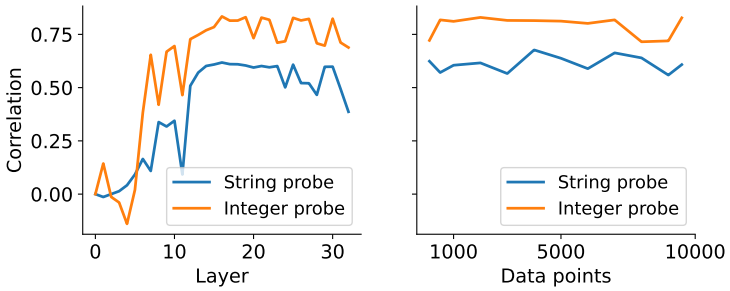

Figure S6: Ablations for the training the probe. The vertical axis shows the correlation between the groundtruth similarities and the probe similarities on the test set. The plot on the left features how the correlation varies with the layer the probe is trained on using the full 9,500 data points for training. The right plot illustrates how the number of data points affects the correlation for a probe on layer 25.

Figure S7A. Here too we found highly structured matrices with patterns that are consistent with those derived from the Levenshtein distance. Quantitatively, we found that all bases resulted in a significant Levenshtein contribution (Figure S7B), and in the case of GPT-4o and Llama-3.1-70b those contributions were enhanced relative to the Log-Linear metric. Indeed, compare the R^2 CIs for the Log-Linear vs. Levenshtein regressors: In the case of Llama-3.1-70b, for base 10 we had: Log-Linear: $[.780, .783]$ and Levenshtein: $[.108, .112]$, whereas for base 4 this is reversed: Log-Linear: $[.336, .340]$, and Levenshtein: $[.366, .371]$. Likewise, for GPT-4o in base 10: Log-Linear: $[.426, .430]$, and Levenshtein: $[.359, .364]$, whereas for base 4: Log-Linear: $[.295, .299]$, and Levenshtein: $[.424, .428]$.

I.2 Number of Comparisons

Within the main paper, we restricted the similarity judgments to a maximum of 1000×1000. Here we extend this for GPT-4o to the 2000×2000, presented in Figure S8. We find that the similarity pattern continues after the number 1000, but begins to get broader, possibly because of tokenization effects.

I.3 Other Qualifiers

Throughout the paper, we used the "similar" qualifier when asking the model to compare two numbers. This is because "similar" is inherently neutral and is better motivated from the psychological literature for exploring representations. As an exploratory analysis we recomputed the GPT-4o similarity matrix using an alternative "close" qualifier which has a more quantitative connotation (similar to the `int()` type). We provide a side-by-side comparison in Figure S8. As expected, we observe that

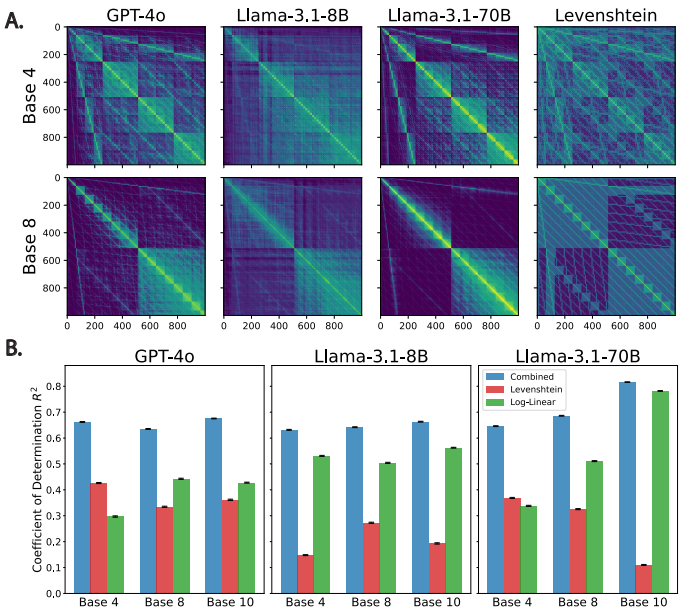

Figure S7: The effect of other number bases on elicited similarity. **A.** LLM similarity matrices over all integer pairs in the range $0 - 999$ represented in base 4 and 8 along with the corresponding Levenshtein distance measures (see Appendix D for prompts). **B.** Coefficient of determination (R^2) for the various similarity matrices under the different base contexts (including the base 10 results from Figure 1) for the combined and separate Levenshtein (string) and Log-Linear (numerical) distance predictors (error bars indicate 95% CIs).

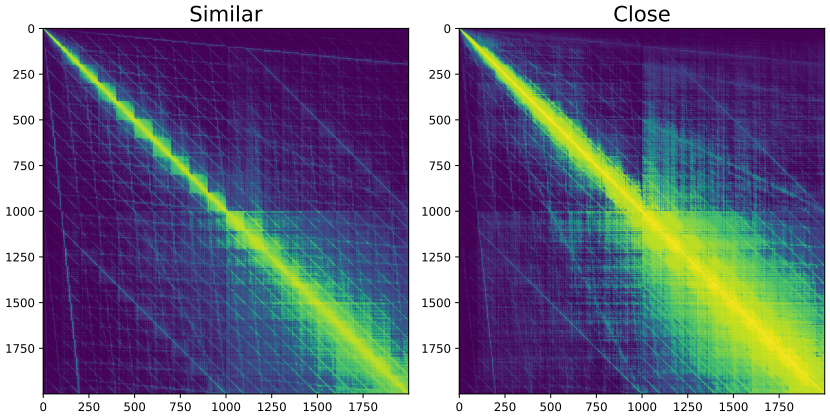

Figure S8: (left) Similarity judgments extended to the 2000×2000 setting. (right) Same prompt but "similar" is replaced with "closer" in the 2000×2000 setting. Both plots are for GPT-4o.

the "close" case has an effect that more closely aligns with the `int()` context. Nevertheless, we still notice a pronounced set of string-induced diagonals, especially as the integers get larger.