Minimization of Boolean Complexity in In-Context Concept Learning

Leroy Z. Wang
Independent
wang.leroy@outlook.com

R. Thomas McCoy
Department of Linguistics
Yale University
tom.mccoy@yale.edu

Shane Steinert-Threlkeld
Department of Linguistics
University of Washington
shanest@uw.edu

Abstract

What factors contribute to the relative success and corresponding difficulties of in-context learning for Large Language Models (LLMs)? Drawing on insights from the literature on human concept learning, we test LLMs on carefully designed concept learning tasks, and show that task performance highly correlates with the Boolean complexity of the concept. This suggests that in-context learning exhibits a learning bias for simplicity in a way similar to humans.

1 Introduction

The human conceptual apparatus represents one of the most remarkable aspects of our species' intelligence [Murphy, 2002]. In order to understand the ways in which artificial intelligences do and do not resemble our own, understanding their conceptual structure is an important first step.

One prominent tradition argues that human concepts are representations in a *language of thought (LoT)* [Fodor, 1975, Goodman et al., 2015, Quilty-Dunn et al., 2022]. Support for this perspective comes from Bayesian arguments that concept learning exhibits a strong bias for simplicity, consistent with a view in which human learners infer the simplest expression in an LoT that is consistent with the data that they have seen [Feldman, 2000, Chater and Vitányi, 2003, Goodman et al., 2008, Piantadosi et al., 2016].

In this paper, we study *in-context concept learning* with large language models (LLMs), allowing us to address the following questions: when presented with labeled examples of an unknown concept, can an LLM infer the underlying concept? If so, what inductive biases does this in-context concept learning exhibit; in particular, does it exhibit a simplicity bias akin to the simplicity bias displayed by humans?

Consider the prompt in (1). In the first two lines, we see labeled examples of a new numerical concept, *bnik*. The final line asks a model to label a new example. Repeating this for a range of example sets and concepts, we can measure whether models have greater success with simpler concepts.

(1) There are 10 apples. Alice has 3 of the apples. Does Alice have bnik of the apples? No.
There are 15 apples. Alice has 10 of the apples. Does Alice have bnik of the apples? Yes.
There are 20 apples. Alice has 10 of the apples. Does Alice have bnik of the apples? ___

We study a range of numerical concepts expressed in a simple language of thought with basic logical and arithmetical operators and find (for several different models) that concepts with shorter representations in a hypothesized LoT are easier to learn in-context. This shows that LLMs are capable of learning non-trivial mathematical concepts and exhibit learning biases that are similar to those used in human concept learning.

2 Related Work

Feldman [2000] showed that ease of human concept learning is highly negatively correlated with Boolean logical complexity: concepts with longer minimal logical formulas are harder for people to learn. A large body of subsequent work has extended the range and scope of this view using Bayesian inference in various LoTs [Goodman et al., 2008, Piantadosi et al., 2016]. Carcassi and Szymanik [2022] show that neural networks trained from scratch to learn Boolean concepts exhibit a similar bias for simplicity. A wide range of work has recently analyzed when, how, and why in-context learning (ICL) in LLMs works [Min et al., 2022, Akyürek et al., 2022, Akyürek et al., 2024, i.a.]. To the best of our knowledge, ours is the first to explicitly study concept learning and measure a learning bias for logical simplicity in ICL.

3 Methodology

3.1 Concept generation

Our data generation methodology is inspired by van de Pol et al. [2022] and Z. Wang and Steinert-Threlkeld [2023], where we define the complexity of a concept using its minimal description length—the length of the shortest expression that can capture the concept (defined more precisely below). We define a concept as a semantic object generated by a logical grammar, whose basic structure is shown in Table 1. The full grammar used during generation is given in Appendix A.4; it imposes some additional constraints to prevent certain types of unwanted recursive generation.[1]

Operator	**Type**	**Gloss**
$=$	NUM $\times$ NUM $\rightarrow$ BOOL	Numerical equality
$\neq$	NUM $\times$ NUM $\rightarrow$ BOOL	Numerical inequality
$>$	NUM $\times$ NUM $\rightarrow$ BOOL	Numerical more than
$<$	NUM $\times$ NUM $\rightarrow$ BOOL	Numerical less than
$\times$	NUM $\times$ NUM $\rightarrow$ NUM	Numerical multiplication
$\wedge$	BOOL $\times$ BOOL $\rightarrow$ BOOL	And
$\vee$	BOOL $\times$ BOOL $\rightarrow$ BOOL	Or

Table 1: Operators in the logical grammar.

The complexity of a concept is determined by the number of operators in its minimal description. For example, the concept $[\![$between 5 and 10$]\!]$ has a complexity of 3 (and therefore will be in *class 3*) because there are three operators ($>$, $\wedge$, and $>$) in its minimal description: $(x > 5) \wedge (10 > x)$. One example of a concept with a complexity of 1 is $[\![$less than 5$]\!]$, with minimal description: $(x < 5)$. Concept complexity class n will thus contain all unique concepts with minimal description lengths of n that can be generated by the logical grammar. The complexity classes, along with some representative concepts for each class, are included in Appendix A.8.

Under the procedure described so far, it is possible to generate two concepts that have similar or identical meanings. Here, concepts classify pairs of numbers (in (1), the total number of apples and the number of apples that Alice has) into true and false. We identify a concept's meaning as its extension, i.e. the set of such pairs that it maps to true.

This fact creates several issues for interpreting the results:

1. If two concept descriptions in different complexity classes yield the same meaning (e.g., $x < 5 \wedge x < 6$ and $x < 5$), then the more complex one should not be considered because

[1] See Appendix A.4 for an example.

our hypotheses are about a concept's *minimal* description length; thus, only the simplest description for a concept should be considered.

2. If two concept descriptions in the same complexity class yield the same meaning (e.g., $x < 5 \vee x > 17$ and $x > 17 \vee x < 5$), then they are effectively the same concept. Thus, generating both concepts is effectively the same as sampling one concept twice, which is undesirable because it might make that concept overrepresented, biasing the results.
3. If two concept descriptions yield similar but non-identical meanings, there are potential challenges in interpreting a model's performance. Consider the concepts with meanings $(x > 5)$ and $(x > 5) \wedge (x \neq 7)$. The second meaning only differs from the first at exactly one place (when $x = 7$). Thus, if we intend to test learning of the second, more complex concept, a model would get almost the same accuracy if it had learned the incorrect simpler concept as it would get if it had learned the correct concept, since both concepts almost always yield identical predictions. This is a problem because it makes it challenging to determine if the model has learned the intended concept (as opposed to a similar but unintended concept).

To address these issues, we perform deduplication in which a generated concept is discarded if its meaning is the same as, or similar to, a previously-generated concept. Since concepts are generated in order of increasing complexity, this procedure ensures that we keep only the minimal description for a given meaning. For deduplication across complexity classes, we consider two concepts similar when their Levenshtein distance[2] is less than 3, i.e. they differ in truth value on at most 3 inputs, in which case we discard the one with the longer description. Deduplication is also performed within complexity classes: when multiple concepts in the same class have a Levenshtein distance of 0 with each other, we discard all but one of them.

3.2 Data generation

Each prompt that we give to LLMs, such as the prompt shown in (1) above, is made of several examples. Each example is generated from the following template, where the slots that vary between examples are underlined. See Appendix A.3 for more details.

(2) There are $\underset{\text{TOTAL}}{\underline{\hspace{1.5cm}}}$ $\underset{\text{OBJ}}{\underline{\hspace{1.5cm}}}$. $\underset{\text{SUBJ}}{\underline{\hspace{1.5cm}}}$ $\underset{\text{PRED}}{\underline{\hspace{1.5cm}}}$ $\underset{\text{NUM}}{\underline{\hspace{1.5cm}}}$ of the $\underset{\text{OBJ}}{\underline{\hspace{1.5cm}}}$.
Does $\underset{\text{SUBJ}}{\underline{\hspace{1.5cm}}}$ $\underset{\text{PRED}}{\underline{\hspace{1.5cm}}}$ bnik of the $\underset{\text{OBJ}}{\underline{\hspace{1.5cm}}}$? $\underset{\text{YES/NO}}{\underline{\hspace{1.5cm}}}$.

3.3 Models

We ran experiments on two LLM families: Qwen2 [Yang et al., 2024] from Alibaba research, and Gemma 2 [Riviere et al., 2024] from Google DeepMind. During testing, the instruction-tuned versions of the models and default Huggingface chat templates were used. Qwen2-72b was the best-performing open model on Hugging Face Open LLM Leaderboard[3] as of June 2024. The Gemma 2 models achieved state-of-the-art results for their size, while reaching competitive performance on many benchmarks when compared to models with $2\times$ more parameters.

4 Results

For each complexity class, we randomly sample 18 concepts from all possible concepts in that class. Results are shown in Figure 1, in which each data point represents the model's accuracy on a specific concept. The trend line shows the line of best fit to the average accuracy of each complexity class.

For each model family, we run experiments for models in two sizes—the largest model in that family, and a model that has approximately 10 billion parameters. As shown in the figure, the average accuracy for all LLMs decreases as concept complexity increases. The drop in average accuracy is most evident in Gemma 2-9B: from 83% in class 1 to 66% in class 5. For the largest model we tested, Qwen2-72B, there is a 16% decrease (90% $\rightarrow$ 74%) in average accuracy as complexity increases from 1 to 5. See Appendix A.2 for a plot of only average accuracy values for each model.

[2] See Appendix A.5 on how the distance is calculated.

[3] `https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard`

In Table 2, we see there is a strong negative correlation between concept complexity and average model accuracy, indicated by the Pearson correlation coefficients (PCC). All results are statistically significant except for Qwen2-72B, which is nearly significant at a $p = 0.05$ threshold. This suggests that the bias for simplicity in in-context learning may become smaller as a function of model size.

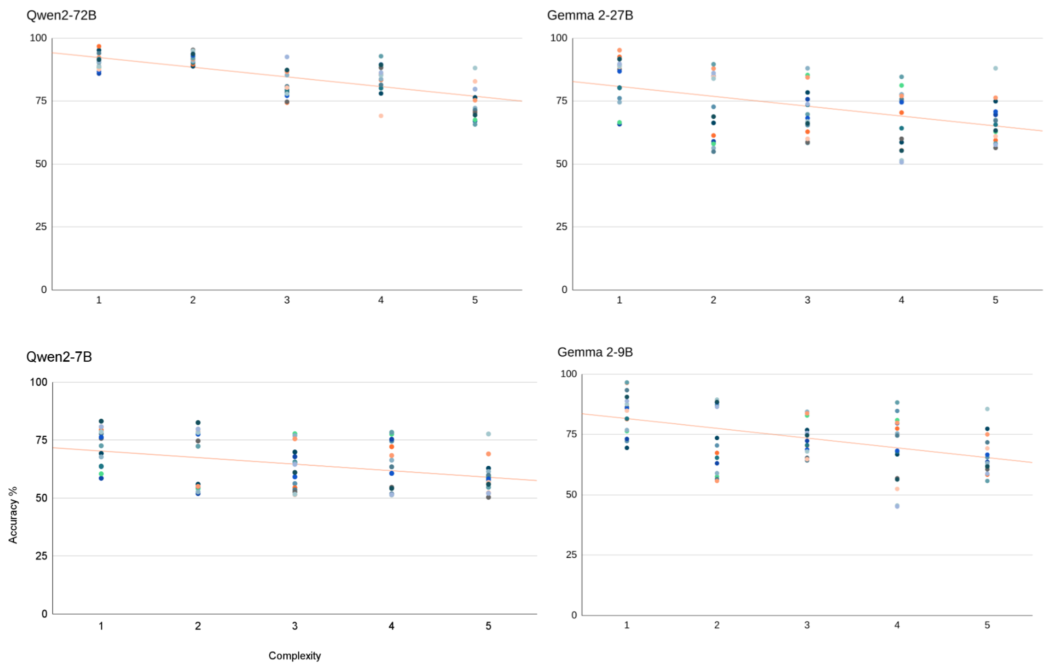

Figure 1: The influence of concept complexity on LLM accuracy at concept learning (see text for how complexity is operationalized). On average, LLM accuracy drops as complexity increases. Each point shows accuracy on a single concept; the lines are lines of best fit to the average accuracy for each complexity class.

Model Name	**PCC**	p**-value**
Gemma-2-**9B**-it	-0.961	0.009
Gemma-2-**27B**-it	-0.898	0.038
Qwen2-**7B**-Instruct	-0.884	0.046
Qwen2-**72B**-Instruct	-0.854	0.065

Table 2: Pearson correlation between complexity and average accuracy

5 Conclusion

This work has shown that LLM in-context concept learning exhibits a simplicity bias of a similar sort as the simplicity bias that has been observed in human concept learning. Much work remains for the future; important future directions include (i) more detailed comparisons with human concept learning data (e.g., of exact learning curves), (ii) extending this research to conceptual domains beyond the numerical, and (iii) more detailed analysis of factors explaining the ease of in-context concept learning beyond LoT complexity. We hope that this work will spur a new line of research exploring how perspectives from the concept learning literature in cognitive science can shed light on the nature of in-context learning in LLMs.

References

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 787–812. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/akyurek24a.html`.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models. 2022. URL `https://openreview.net/forum?id=0g0X4H8yN4I`.

Fausto Carcassi and Jakub Szymanik. Neural networks track the logical complexity of boolean concepts. *Open Mind : Discoveries in Cognitive Science*, 6:132 – 146, 2022. URL `https://direct.mit.edu/opmi/article/doi/10.1162/opmi_a_00059/112493/Neural-Networks-Track-the-Logical-Complexity-of`.

Nick Chater and Paul Vitányi. Simplicity: A unifying principle in cognitive science? *Trends in Cognitive Sciences*, 7(1):19–22, 2003.

Jacob Feldman. Minimization of Boolean Complexity in Human Concept Learning. *Nature*, 407:630–633, 2000. doi: 10.1038/35036586.

Jerry A. Fodor. *The Language of Thought*. Harvard University Press, 1975. ISBN 978-0-674-51030-2.

Noah D Goodman, Joshua B Tenenbaum, Jacob Feldman, and Thomas L Griffiths. A rational analysis of rule-based concept learning. *Cognitive Science*, 32(1):108–154, 2008. doi: 10.1080/03640210701802071.

Noah D. Goodman, Joshua B. Tenenbaum, and Tobias Gerstenberg. Concepts in a Probabilistic Language of Thought. In Eric Margolis and Stephen Laurence, editors, *The Conceptual Mind*, pages 623–654. The MIT Press, 2015. ISBN 978-0-262-32686-5. doi: 10.7551/mitpress/9383.003.0035. URL `https://direct.mit.edu/books/book/3430/chapter/115711/Concepts-in-a-Probabilistic-Language-of-Thought`.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.759. URL `https://aclanthology.org/2022.emnlp-main.759`.

Gregory L. Murphy. The big book of concepts. 2002. URL `https://mitpress.mit.edu/9780262632997/the-big-book-of-concepts/`.

Steven T Piantadosi, Joshua B Tenenbaum, and Noah D Goodman. The logical primitives of thought: Empirical foundations for compositional cognitive models. *Psychological Review*, 123(4):392–424, 2016. doi: 10.1037/a0039980.

Jake Quilty-Dunn, Nicolas Porot, and Eric Mandelbaum. The Best Game in Town: The Re-Emergence of the Language of Thought Hypothesis Across the Cognitive Sciences. pages 1–55, 2022. ISSN 0140-525X, 1469-1825. doi: 10.1017/S0140525X22002849. URL `https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/best-game-in-town-the-reemergence-of-the-language-of-thought-hypothesis-across-the-cognitive-s` `76F46784C6C07FF52FF45B934D6D3542`.

Gemma Team Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, L'eonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ram'e, Johan Ferret, Peter Liu, Pouya Dehghani Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stańczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda

Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Boxi Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Christoper A. Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozi'nska, D. Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Pluci'nska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost R. van Amersfoort, Josh Gordon, Josh Lipschultz, Joshua Newlan, Junsong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, L. Sifre, L. Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Gorner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, S. Mc Carthy, Sarah Perrin, S'ebastien Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomás Kociský, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Brian Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeffrey Dean, Demis Hassabis, Koray Kavukcuoglu, Cl'ement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size. *ArXiv*, abs/2408.00118, 2024. URL `https://arxiv.org/pdf/2408.00118`.

Iris van de Pol, P. Lodder, Leendert van Maanen, Shane Steinert-Threlkeld, and Jakub Szymanik. Quantifiers satisfying semantic universals have shorter minimal description length. *Cognition*, 232, 2022. URL `https://www.sciencedirect.com/science/article/pii/S001002772200138X`.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. Qwen2 technical report. *ArXiv*, abs/2407.10671, 2024. URL `https://arxiv.org/abs/2407.10671`.

Leroy Z. Wang and Shane Steinert-Threlkeld. GQG: Generalized quantifier generalization - a dataset for evaluating quantifier semantics understanding in language models. In Dieuwke Hupkes, Verna Dankers, Khuyagbaatar Batsuren, Koustuv Sinha, Amirhossein Kazemnejad, Christos Christodoulopoulos, Ryan Cotterell, and Elia Bruni, editors, *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, pages 185–192, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.genbench-1.15. URL `https://aclanthology.org/2023.genbench-1.15`.

A Appendix / supplemental material

A.1 Limitations

Only a finite set of fractions are used in the grammar to improve efficiency. As a consequence, it may be the case that some concepts that require a certain minimal number of operators under our framing could be expressed using fewer operators if more fractions were allowed.. To address this issue, we plan to use a richer set of fractions in future work.

Only a limited set of LLMs are tested. It is possible that newer models / models with different architectures do not exhibit the phenomena discussed in this text.

We use only one prompt template for all experiments, which may introduce implicit biases in the data and affect the experiment results.

A.2 Complexity vs. average accuracy plot

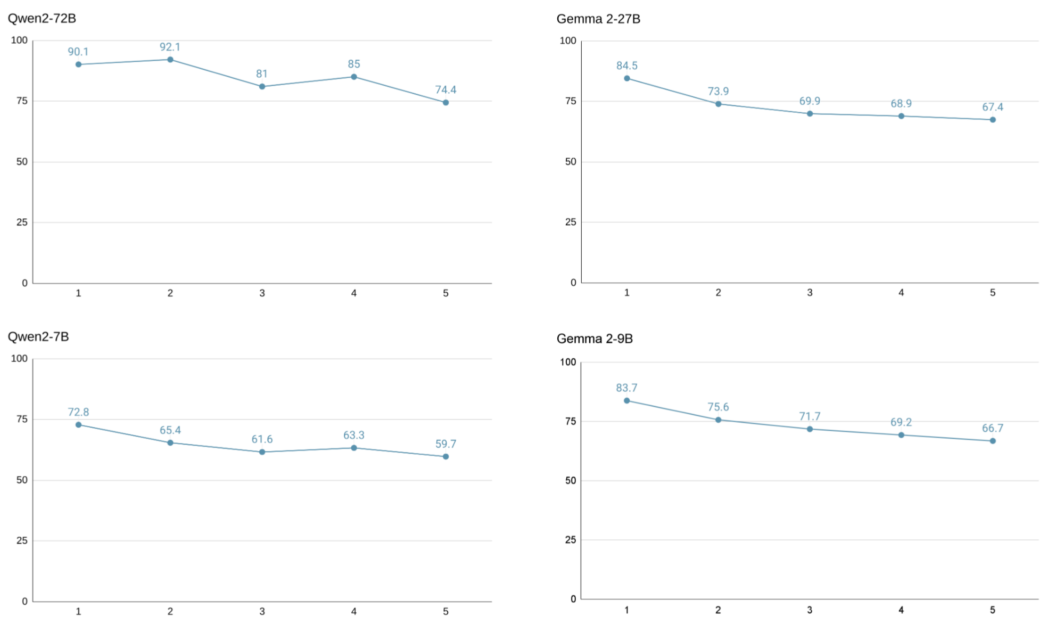

Figure 2: Complexity vs. average accuracy.

A.3 Data Generation

The pseudocode shows how prompt data is generated, from sets of (predefined) subjects, verbs, and objects.

The sets of examples used in this paper are balanced – in every prompt, the model sees the same number (10) of positive and negative examples[4]; and for each accuracy data point, the model is tested on the same number (500) of prompts with true labels and false labels. In this work, we use the template "*Let us define a new word, bnik.*", followed by labeled examples and a question at the end, for all prompts in the dataset.

[4]Because of this, we exclude concepts that would have fewer than 10 positive or negative examples.

Algorithm 1 Data Generation

```
Inputs: set of subject nouns S, set of predicate verbs P, set of objects O,
function GENERATE_EXAMPLE

initialize positive_examples = [ ], negative_examples = [ ]

for total in [5, 100] do
    for num in [0, total] do
        uniformly randomly sample s, p, o from S, P, O
        example = GENERATE_EXAMPLE (total, num, s, p, o)

        append example with true labels to positive_examples
        append example with false labels to negative_examples
    end for
end for

return positive_examples, negative_examples
```

As shown in Algorithm 1 in Appendix A.3, we iterate through all meaningful numerical ranges for both the number of total objects (which we restrict to be between 5 and 100 inclusive) and the number of objects the person has, and generate an example for each combination. If we want to generate a prompt with m positive examples and n negative examples, we would sample without replacement m and n examples from the sets of positive examples and negative examples respectively, and use an unseen example as a question at the end. In our experiments, the prompt always shows the same number of negative examples as positive ones—that is, m and n are equal.

A.4 Grammar for concept generation

The grammar used during concept generation is given below, presented as a context-free grammar.

```
Bool -> Bool   and   Bool
Bool -> Bool   or    Bool

Bool -> Var2   ==    SimpleInt
Bool -> Var2   !=    SimpleInt
Bool -> Var2   >     SimpleInt
Bool -> Var2   <     SimpleInt

Bool -> Var2   >     ComplexFloat
Bool -> Var2   <     ComplexFloat


ComplexFloat -> SimpleFloat   *   Var1


# total number of items
Var1 -> [5, 100]

# subject's number of items
Var2 -> [0, 100]

SimpleInt -> [0, 100]

# fractions
SimpleFloat -> {1/5, 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5}
```

Constraints on the grammar For the sake of efficiency, the fine-grained type system in the grammar prevents the generation of fractions that are not in the predefined list of fractions. Multiplication can only take place between "SimpleFloat" and "Var1", so it is not possible to obtain new fractions by multiplying existing ones.

A.5 Deduplication method

The deduplication methodology used here is based on vectors representing concept meanings. Suppose the total number of objects is $n = 100$. We can then imagine the semantics of a concept being represented by a 101-dimensional vector, where each dimension is the truth value of $f(n = 100, x) = \{\langle n, x\rangle : (x > 3)\}$ for $x = [0, 100]$.

For instance,

```
x=0    (x > 3) False    0
x=1    (x > 3) False    0
x=2    (x > 3) False    0
x=3    (x > 3) False    0
x=4    (x > 3) True     1
x=5    (x > 3) True     1
......

-> [0 0 0 0 1 1 ......].
```

We compute such a semantics vector for $n = \{25, 50, 100\}$ for all concepts in each class, and use the edit distance[5] to remove similar concepts. Two concepts are considered similar if they belong to different classes and have an edit distance < 3 between their vectors.

Algorithm 2 Concept deduplication: when two concepts are similar, only remove the one in the more complex class

---

Inputs: current concept class $this\_class$, set of all previous concept classes $prev\_classes$
function EDIT_DIST

$deduped\_concepts = this\_class$

for $concept$ in $this\_class$ **do**
  for $prev\_concept$ in $prev\_classes$ **do**
    if EDIT_DIST ($concept$, $prev\_concept$) < 3 **then**
      remove $concept$ from $deduped\_concepts$
    end if
  end for
end for

return $deduped\_concepts$

---

[5]Since any two concept vectors always have the same length, the distance is defined as the number of places where two vectors have different values.

A.6 Example prompt

Prompt	Label
Let us define a new word, bnik. There are 17 plants. Alice has 13 of the plants. Does Alice have bnik of the plants? No. There are 99 trees. Bob has 7 of the trees. Does Bob have bnik of the trees? Yes. There are 40 tables. Alice owns 36 of the tables. Does Alice own bnik of the tables? No. There are 72 chairs. Bob owns 9 of the chairs. Does Bob own bnik of the chairs? Yes. There are 82 chairs. Bob has 47 of the chairs. Does Bob have bnik of the chairs? No. There are 100 plants. Alice owns 70 of the plants. Does Alice own bnik of the plants? No. There are 56 chairs. Alice owns 3 of the chairs. Does Alice own bnik of the chairs? Yes. There are 56 birds. Bob owns 37 of the birds. Does Bob own bnik of the birds? No. There are 84 tables. Alice owns 12 of the tables. Does Alice own bnik of the tables? Yes. There are 69 bikes. Alice owns 32 of the bikes. Does Alice own bnik of the bikes? Yes. **There are 99 apples. Alice has 3 of the apples.** **Does Alice have bnik of the apples?**	Yes

Table 3: Example of prompts in the dataset. The underlying concept in this example is “less than half”.

Lexical items used in prompts

- nonce word: *bnik*
- subjects: Alice, Bob
- predicates: has, owns
- objects: "tables", "chairs", "apples", "bikes", "trees", "fish", "birds", "plants"

A.7 Compute resources

All experiments were run on two Nvidia RTX A6000 GPUs. Greedy decoding was used during inference.

A.8 Concept complexity classes

Below are some representative concepts for each class. It is not a comprehensive list of all the concepts that are generated.

Complexity 1:
primitive operators

[n = x]
[n > x]

[x > c]
[x < c]

[x = c]
[x != c]

Complexity 2:
proportional primitives

[x > p * n]
[x < p * n]

Complexity 3:
conjunction / disjunction of primitive operators

(x > c1) or (x < c2)
(x > c1) and (x < c2)

Complexity 4:

conjunction / disjunction of one operator and one proportion

(x > c) or (x < p * n)
(x > c) and (x < p * n)

Complexity 5:

conjunction / disjunction of two proportions

(x > p1 * n) or (x < p2 * n)
(x > p1 * n) and (x < p2 * n)

conjunction / disjunction of three primitives

((x > c1) and (x < c2)) or (x > c3)
((x > c1) and (x < c2)) or (x < c3)