©Copyright 2025
Leroy Wang

Complexity of In-Context Concept Learning in Language Models

Leroy Wang

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2025

Committee:
Shane Steinert-Threlkeld
R. Thomas McCoy

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Complexity of In-Context Concept Learning
in Language Models

Leroy Wang

Chair of the Supervisory Committee:
Shane Steinert-Threlkeld
Department of Linguistics

This thesis studies the factors that contribute to the success and shortcomings of in-context learning, which refers to the ability of some language models to perform a new task during inference using only a few labeled examples, 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 logical complexity of the concept. This suggests that in-context learning exhibits a learning bias for simplicity in a way similar to humans.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

The ability of language models to learn new tasks in-context (Brown et al., 2020) seems both exciting and mysterious to Natural Language Processing (NLP) researchers. What makes such learning algorithms work? In what ways is in-context learning similar / different when compared to the human learning process? And can we utilize techniques from cognitive science to better study the mechanisms of such learning algorithms?

We will investigate the learning biases of in-context learning algorithms of large language models (LLMs) using techniques derived from human concept learning experiments. One prominent tradition argues that concepts are representations in a *language of thought (LoT)* (Fodor, 1975; Goodman et al., 2015; Quilty-Dunn et al., 2022). The recent Bayesian revolution in cognitive science has argued that concept learning exhibits a very strong bias for simplicity: 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; Griffiths et al., 2024).

In this thesis, we study *in-context concept learning* with large language models (LLMs). In-context learning in LLMs refers to language models being able to perform a new task using only a few labeled examples given during inference. The experiments allow 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 similar to the one 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.

Chapter 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, this thesis is the first to explicitly study concept learning and measure a learning bias for logical simplicity in ICL.

Misra et al. (2023) constructed datasets testing how language models handle semantic property inheritance with nonce words. They found that language models can reason about the semantic properties of the nonce words reasonably well, but performance drops significantly when distracting information is added to the input. Marinescu et al. (2024) showed that priors from symbolic Bayesian models can be learned by artificial neural networks, resulting in neural nets that exhibit the same inductive biases (as Bayesian models) in concept learning tasks. Unlike our work that uses large language models and natural language data, this paper focuses on smaller neural networks and artificial data. Steinert-Threlkeld (2021) studies the semantic properties (e.g. semantic universals) of natural language quantifiers using a language generated by a logical grammar, which inspired the construction of the logical grammar used in this work.

Agmon et al. (2019; 2022); Tan et al. (2023) have shown that humans generally require

more processing time and tend to make more mistakes when making judgments about ⟦less than⟧ concepts, relative to the ⟦more than⟧ counterparts. An example experiment (for ⟦more than⟧) can be human participants making truth value judgments for the sentence "*over half of* the students are in the park" given a specific context. The authors hypothesized that such phenomena can be attributed to ⟦less than⟧ being a more complex concept than ⟦more than⟧, therefore requiring more time to process. Such results show that humans may have special learning biases for certain concepts, and it is an interesting direction to test whether language models possess similar biases. We will briefly study this direction toward the end of the thesis.

Chapter 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 3.1. The full grammar used during generation is given in section 3.4.1, which imposes some additional constraints to prevent certain types of unwanted recursive generation. Code for generating the concepts, as well as prompting models and analyzing data, may be found at `https://github.com/lerow/llm-concept-learning-complexity`.

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 3.1: 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: $(x > 5) \wedge (10 > x)$; we call this a minimal description because it is the shortest description in the logical grammar that describes the concept. The description $(x > 4) \wedge (x \neq 5) \wedge (10 > x)$ also describes the same concept, but it is not minimal because there exist shorter possible descriptions. 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 section 3.4.

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 number of apples and the number that Alice has) into true and false. We identify a concept's meaning as its extension, i.e. the set of 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 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 $(x > 5) \wedge (x \neq 7)$. When $n = 100$, 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 since it means we cannot tell 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[1] 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:

(1) There are $\underline{\quad}_{\text{TOTAL}}$ $\underline{\quad}_{\text{OBJ}}$. $\underline{\quad}_{\text{SUBJ}}$ $\underline{\quad}_{\text{PRED}}$ $\underline{\quad}_{\text{NUM}}$ of the $\underline{\quad}_{\text{OBJ}}$.

Does $\underline{\quad}_{\text{SUBJ}}$ $\underline{\quad}_{\text{PRED}}$ bnik of the $\underline{\quad}_{\text{OBJ}}$? $\underline{\quad}_{\text{YES/NO}}$.

[1]See 3.3 on how the distance is calculated.

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, we iterate through all meaningful [2] 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. 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[3]; and for each accuracy data point, the model is tested

[2] i.e. a person cannot have more objects than the total number of objects, or have less than 0 objects

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

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.

3.3 Deduplication

The concept 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[4] to remove similar concepts. Two concepts are considered similar if they belong to different classes and have an edit distance < 3 between their vectors.

Different values of n are used during deduplication since it's possible for two concepts to have the same semantics for some n but different semantics for a different n. Consider

[4]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.

concepts $[\![x > 50]\!]$ and $[\![x > \frac{1}{2} \cdot n]\!]$. They have identical semantics when $n = 100$ but not when $n = 50$.

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*

3.3.1 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 3.2: 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”

The exact set of lexical items used in the prompt is presented in this section. The words used for subjects, predicates, and objects are manually chosen by the author. The set of objects is chosen to represent a variety of scenarios while also minimizing ambiguity. A nonce word that is morphologically uncommon in English is used in the prompt to minimize the effect of any pre-existing bias that may be associated with the nonce words (such as “wugs”, “buba”) used in previous literature. Misra et al. (2023) showed that using different nonce words in LLM prompts did not lead to significant bias, but more experiments are needed to study the effect of different nonce words on model performance in concept learning, which we will leave as future work.

3.4 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

$((x > c1)\, and\, (x < c2))\, or\, (x > c3)$

$((x > c1)\, and\, (x < c2))\, or\, (x < c3)$

3.4.1 Grammar for concept generation

The grammar used during concept generation is given below, presented as a context-free grammar. The structuring of the grammar and the logical operators is inspired by Steinert-Threlkeld (2021). To improve efficiency during concept generation, some specific design decisions are made, such as the distinction between "SimpleFloat" and "ComplexFloat", which is explained below.

```
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 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 variable with type "SimpleFloat" can only take the value as one of the fractions defined

above; a variable with type "SimpleInt" can only take the value of a whole number from 0 to 100.

3.5 Counting the number of concepts

Given an arbitrary n, how many concepts are in class n? van de Pol et al. (2022) claimed that the number of possible quantifiers grows exponentially as the minimal expression length increases, but the authors did not provide detailed justification. Here, we will provide a more formal inquiry into this matter.

Let us start with simpler cases. Suppose $n = 1$, the possible concepts in class 1 must have a minimal expression length (MEL) of 1. To count the number of possible concepts with MEL of 1, we need to find the grammar rules that contain just 1 operator[5] (even after all applicable production rules are applied), and enumerate the concepts that can be generated by those rules.

For $n = 2$, we can follow similar steps and find all rules with 2 operators, which are rules 7 and 8 in our grammar.

For $n = 3$, we see that there are no rules that contain 3 operators, so we will need to use rules that have logical connectives to form compositions of simple rules. Since the binary logical connectives in this grammar have an expression length of 1, the expression of each side of the logical operator must also have a length of 1, otherwise we cannot obtain a length-3 expression. If we let $f(n)$ be the number of possible concepts in class n, then we can write $f(3)$ as

$$\begin{aligned} f(3) &= f(1) \cdot 2 \cdot f(1) \\ &= 2 \cdot f(1) \cdot f(1) \end{aligned}$$

[5]which would be rules 3 – 6 in the above grammar

The constant 2 is in the equation since we have two possible logical operators in the grammar. Because we are only interested in the asymptotic behavior of $f(n)$, small constant terms will be omitted in calculations below.

We can observe that for $n > 2$, the expression must contain at least one logical connective. Thus, for $n > 2$, the concept expression must have the form

$$\text{Bool} \;\; \text{–>} \;\; \text{Bool} \;\; \text{Conj} \;\; \text{Bool} \tag{3.1}$$

where 'Conj' is a binary logical operator.

Since 'Conj' has a length of 1, the sum of the lengths of the expressions on the left and right of 'Conj' must be $n - 1$. If we let l and r be the lengths of those two expressions, we can write

$$\begin{aligned}
(n-1) &= l \quad + \quad r \\
&= 1 \quad + \quad (n-2) \\
&= 2 \quad + \quad (n-3) \\
&= 3 \quad + \quad (n-4) \\
&\ldots\ldots \\
&= (n-2) \quad + \quad 1
\end{aligned}$$

Because the logical operators we have are commutative,

$$\text{Bool} \;\; \text{–>} \;\; \text{A} \;\; \text{Conj} \;\; \text{B}$$

is equivalent as

$$\text{Bool} \;\; \text{–>} \;\; \text{B} \;\; \text{Conj} \;\; \text{A}.$$

So when we are counting the unique combinations of $l + r$, we can omit combinations such as

$$(n-2) \quad + \quad 1$$

since it is equivalent to

$$1 \quad + \quad (n-2).$$

To find the number of all possible expressions with form (3.1) and an expression length of n, we can first enumerate all unique combinations of l and r, and for each combination, enumerate all possible expressions with those lengths. We can have the recurrence relation

$$f(n) = \sum_{i=1}^{\lfloor \frac{n-1}{2} \rfloor} f(i) \cdot f(n-i-1) \tag{3.2}$$

where $f(i)$ is the number of possible expressions with length l and $f(n-i-1)$ counts the number of possible expressions with length r. For each value of i, $f(i) \cdot f(n-i-1)$ counts the number of possible expressions such that $l = f(i)$ and $r = f(n-i-1)$; in other words, it counts all possible expressions that can be formed given the lengths of the expressions that are on the left and the right sides of the logical operator. After finding the number of expressions for each combination of l and r, the numbers are summed up to obtain the total number of possible expressions for length n.

Definition 3.5.1 *The n-th Catalan number is defined by* $C_n = \sum_{i=0}^{n-1} C_i \cdot C_{n-i-1}$, *where* $C_0 = 1$.

To find the asymptotic behavior of $f(n)$, we can use the Catalan numbers, which have a similar recurrence form. We see that $f(n)$ in 3.2 is clearly bounded by

$$C(n) = \sum_{i=0}^{n-1} C(i) \cdot C(n-i-1) \tag{3.3}$$

as n approaches infinity, since $f(n)$ has a smaller upper bound of summation (about half of $C(n)$'s upper bound). We can use the known results of C_n to approximate the behavior of $f(n)$. C_n grows in $O(4^n)$ as n increases[6], so we have

$$f(n) = \sum_{i=1}^{\lfloor \frac{n-1}{2} \rfloor} f(i) \cdot f(n-i-1) \tag{3.4}$$

$$= O(4^n) \tag{3.5}$$

where $f(n)$ grows in the order of 4^n.[7]

3.5.1 Scaling up concept generation via sampling

As demonstrated in section 3.5, scaling up the experiments is not feasible with brute-force enumeration due to the exponential growth in the number of concepts. To address this issue, we will use random sampling to ameliorate the computational bottleneck.

We can formulate our problem as

$$\text{Given an arbitrary } n\text{, find an expression in class } n. \tag{3.6}$$

To solve (3.6), we can break it down into different steps below, each with increasing difficulty.

1. Find an expression with length n using random sampling

[6] `https://mathworld.wolfram.com/CatalanNumber.html`

[7] A more precise asymptotic analysis is possible, and can be found in literature studying Catalan numbers. We omit the details as it is not the focus of this section.

2. Make sure each expression in class n is equally likely to be sampled (i.e. no bias)

3. Find conditions that guarantee the expression length is minimal (or not minimal) when satisfied.

For step 1, we can exploit the fact that expressions must have the structure (3.1) when $n > 2$. When $n = 1$ or $n = 2$, the problem is trivial since we can just randomly sample from the production rules that contain 1 or 2 operators. For larger n, we can follow similar steps shown in 3.5, and let l and r be the lengths of the two expressions on the left and right of the logical operator:

$$\begin{aligned}(n-1) &= l \quad + \quad r \\ &= 1 \quad + \quad (n-2) \\ &= 2 \quad + \quad (n-3) \\ &= 3 \quad + \quad (n-4) \\ &\ldots\ldots\end{aligned}$$

We want to find all such combinations of l and r given that $l \leq r$. Once we have the set of combinations, we can uniformly randomly sample a combination (l, r) from the set. If from the sampled combination $l > 2$, then we will repeat the same steps by finding all l_2, r_2 such that $(l-1) = l_2 + r_2$ and $l_2 \leq r_2$. We repeat the same steps for r, until all l, r are not greater than 2.

To demonstrate using an example, suppose $n = 8$. We have

$$\begin{aligned}(8-1) &= l + r \\ &= 1 + 6 \\ &= 2 + 5 \\ &= 3 + 4.\end{aligned}$$

From the three pairs of (l, r), suppose we randomly sampled $(2, 5)$. We continue the procedure for r since it is greater than 2.

$$\begin{aligned}(5-1) &= l + r \\ &= 1 + 3 \\ &= 2 + 2.\end{aligned}$$

Now suppose we sampled $(1, 3)$, we will continue with $r = 3$:

$$\begin{aligned}(3-1) &= l + r \\ &= 1 + 1.\end{aligned}$$

Figure 3.1 visualizes this procedure below.

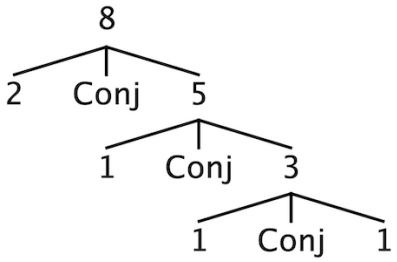

Figure 3.1: Example of random sampling an expression with length 8.

A possible expression ($n = 8$) that can be generated using the above procedure, **where T is the total number of items**:

$$(x > \frac{1}{2} \cdot T) \ \text{AND} \ [(x > 8) \ \text{OR} \ [(x > 1) \ \text{AND} \ (x < 5)]]. \tag{3.7}$$

For step 2, we need to ensure that our sampling procedure is not biased toward any particular structure. Note that the sampling procedure presented above may not be unbiased, since we cannot be certain if the structures[8] `(2 Conj 5)` and `(3 Conj 4)` have the same number of possible expressions, yet each structure is sampled uniformly randomly.

To address this issue, we can let the sampling step be weighted, such that each structure is sampled with a probability proportional to the number of all possible expressions with that structure. Using the $n = 8$ example,

[8] where number a represents a length-a expression

$$
\begin{aligned}
(8-1) &= l + r \\
&= 1 + 6 \quad \text{with probability} \quad f(1) \cdot f(6)/t \\
&= 2 + 5 \quad \text{with probability} \quad f(2) \cdot f(5)/t \\
&= 3 + 4 \quad \text{with probability} \quad f(3) \cdot f(4)/t.
\end{aligned}
$$

where

$$t = f(1) \cdot f(6) \; + \; f(2) \cdot f(5) \; + \; f(3) \cdot f(4).$$

For step 3, which poses the greatest difficulty, we will prove a few conditions that guarantee an expression is not minimal when satisfied.

Lemma 3.5.2 *For a fixed T, let e be an expression that contains only one* `AND` *and no other logical operators. e is not minimal if it does not have the form $a < x < b$ where $b > a$.*

Proof of 3.5.2 If e does not have the form ($x > a$ AND $x < b$) such that $b > a$, then it must be the case that

1. e has the form ($x > a$ AND $x > b$), or ($x < a$ AND $x < b$); the former is equivalent[9] as ($x > c$) which is a shorter expression, where c is the larger number between a and b; the latter is equivalent as ($x < c$), where c is the smaller number between a and b. Therefore e is not minimal.

2. or, we have ($x > a$ AND $x < b$) where $b \leq a$, which is an expression that is always false. Therefore e is not minimal.

[9] "extensionally equivalent"

Lemma 3.5.3 *For a fixed T, let e be an expression that contains only one* `OR` *and no other logical operators. e is not minimal if it does not have the form ($x < a$ OR $x > b$) where $b > a$.*

Proof of 3.5.3 If e does not have the form ($x < a$ OR $x > b$) such that $b > a$, then we must have

1. e has the form ($x > a$ OR $x > b$), or ($x < a$ OR $x < b$); the former is equivalent as ($x > c$) which is a shorter expression, where c is the smaller number between a and b; the latter is equivalent as ($x < c$), where c is the larger number between a and b. Therefore e is not minimal.

2. or, we have ($x < a$ OR $x > b$) where $b \leq a$, which is an expression that is always true. Therefore e is not minimal.

Future work Previous works such as van de Pol et al. (2022) and Wang et al. (2024) limit the max length of expression to 5 due to the exponential growth of generated concepts. This section sets the basic theoretical framework for scaling up concept generation using random sampling. We note that the non-minimality conditions presented in step 3 are nascent and only work in ideal scenarios (e.g. for a fixed total number). Some next steps to improve these conditions can be

1. to show that an expression e containing multiple logical connectives is not minimal, if at least one clause in e (connected to other clauses by a logical connective) is not minimal. This should not be difficult to prove, since one can show that the non-minimal clause can be replaced by a shorter expression while keeping the extensional semantics of e the same.

2. to show that if an expression does not satisfy the non-minimality conditions, then it must be minimal. This can be difficult to prove since expressions about proportions

(e.g. $x > \frac{1}{2} \cdot T$) can have different truth conditions for different T. An easier statement to prove might be "suppose e is an expression with multiple clauses connected by logical operators, and fix the value of T to be the largest valid number in the grammar, e is non-minimal at T if the truth conditions[10] of any two clauses are not disjoint. " The statement might be a sufficient but not necessary condition for non-minimality, which could be useful for efficiently finding longer expressions with minimality guarantees. Note that the statement (along with its inverse) is not true, if the grammar contains a modulo operator % with type
NUM $\times$ NUM $\rightarrow$ NUM, then $(x \,\%\, 2 = 0)$ would be extensionally equivalent as $(x = 0)$ OR $(x = 2)$ OR $(x = 4)$ OR $(x = 6)$ OR ...$(x = d)$, where d is the largest valid even number in the grammar.

[10]see 3.3 for details on expression truth conditions

Chapter 4

RESULTS

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[1] 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× more parameters.

For each complexity class, we randomly sample 18 concepts from all possible concepts in that class. Each data point on the plot in Figure 4.1 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. Figure 4.2 shows models' average accuracies for the same set of experiments.

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.

In Table 4.1, 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.

---

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

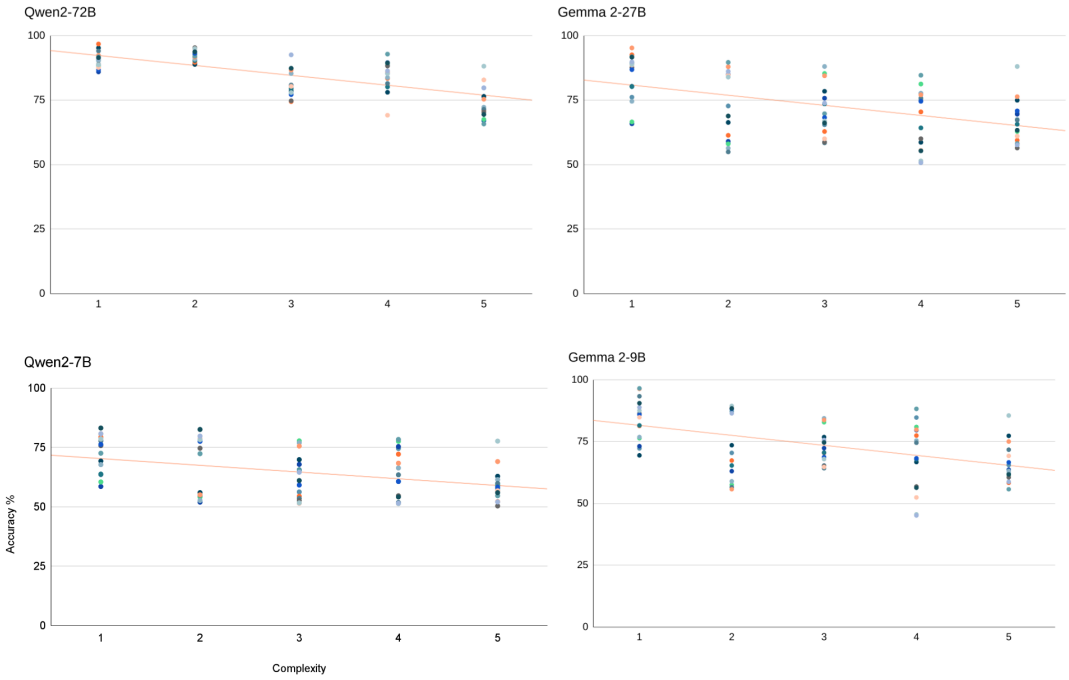

Figure 4.1: The influence of concept complexity on LLM accuracy in concept learning. On average, LLM accuracy drops as complexity increases. Each data point in the plot represents the model's accuracy on a specific concept. The orange-colored trend line shows the line of best fit to the average accuracy of each complexity class.

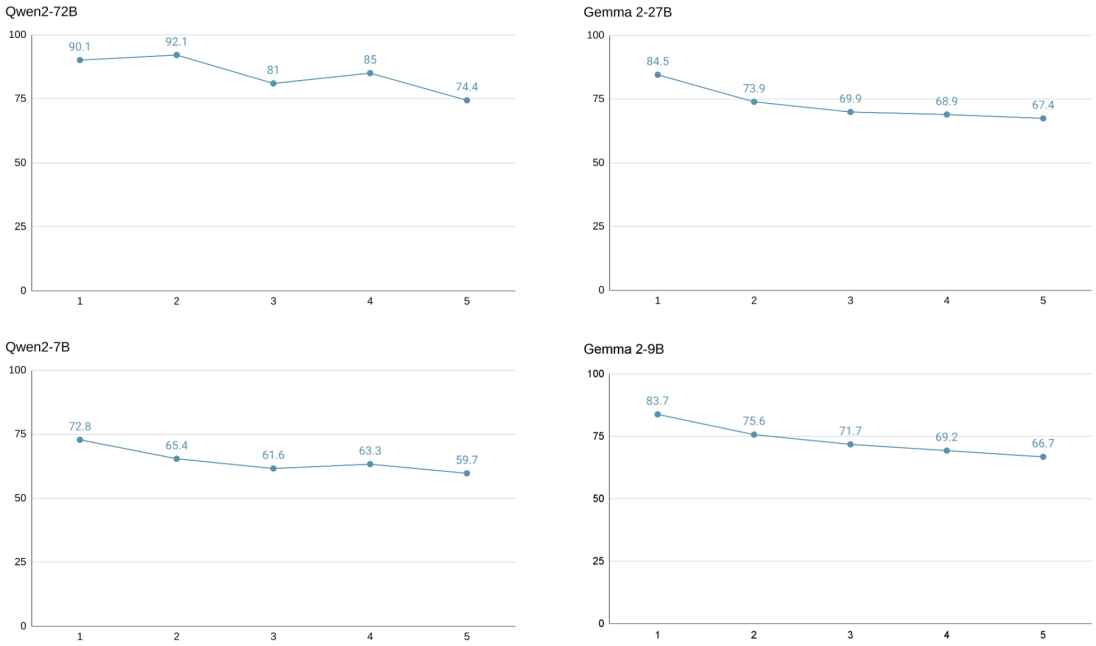

Figure 4.2: The influence of concept complexity on LLM accuracy in concept learning. Each data point in the plot represents the model's *average* accuracy for the complexity class.

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 4.1: Pearson correlation between complexity and average accuracy

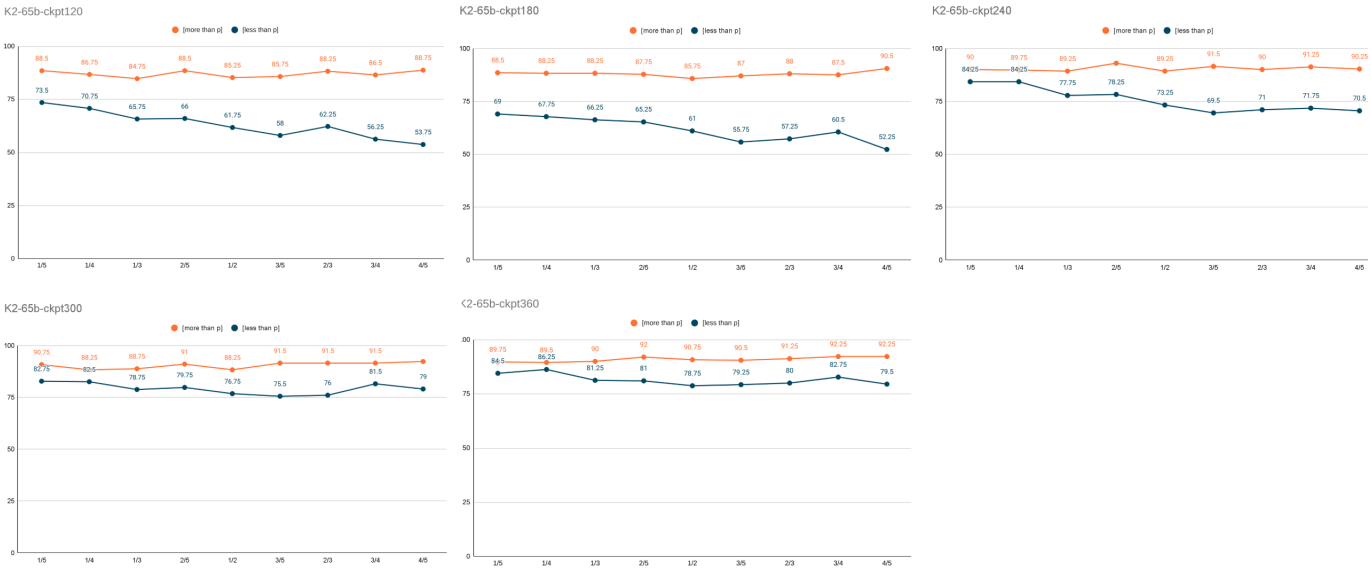

Figure 4.3: LLM accuracy of learning concepts ⟦more than⟧ vs ⟦less than⟧ at different model checkpoints (checkpoints 120, 180, 240, 300, 360). The model reaches higher accuracies for ⟦more than⟧ across all checkpoints. At earlier checkpoints, the accuracy gap between the two concepts tends to be larger.

More vs less in K2-65b language model Figure 4.3 shows the results of comparing the learning accuracies between concepts ⟦more than p⟧ and ⟦less than p⟧ (where p is a proportion) in K2-65b model (Liu et al., 2025). Each plot in the figure represents the result for the same set of experiments obtained from each checkpoint. The x-axis of the plot shows the different values of proportion p, and the y-axis is the accuracy. For each value of p, the same random sampling procedure (used in experiment 4.1) is used to generate 200 positive examples (i.e. with true label "yes") and 200 negative examples. After a fixed number of training steps, a model checkpoint is saved for K2-65b. A larger checkpoint number means the checkpoint is from a later training stage of the model. For reference, K2-65b has 380 checkpoints (numbered from 001 to 380) in total.

We see that generally ⟦less than p⟧ is a harder concept to learn compared to ⟦more than p⟧,

since on average K2-65b has a lower accuracy for ⟦less than p⟧ across all checkpoints. This can have many interpretations; (Agmon et al., 2019; 2022) have shown that humans generally need more processing time and can make more mistakes when making judgments about ⟦less than⟧ concepts, compared to the ⟦more than⟧ counterparts. They argued that ⟦less than⟧ is a concept that contains an implicit negation, which requires more processing time. In other words, they claimed that ⟦less than⟧ is potentially being computed as ⟦NOT more than⟧ in humans; since there is one more operator to compute, ⟦less than⟧ becomes more computationally expensive. This argument can be a possible explanation for this phenomenon in K2-65b. Although the two concepts have the same logical complexity in our current grammar, we can modify the grammar (e.g. removing the $>$ operator) so that ⟦less than p⟧ is always more complex than ⟦more than p⟧. The advantage of this approach is that this phenomenon can be explained using the main results (complexity vs accuracy) of the thesis – ICL tends to favor logical simplicity. However, it also means that the main experiments need to be re-run using the new grammar, which can lead to different outcomes compared to our current results. Another approach is to keep the grammar unchanged, and argue that there can be other factors besides logical complexity that affect the learning accuracy. We leave this as an exciting direction for future work.

4.1 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 or 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.

Chapter 5
CONCLUSION

We demonstrate that LLM in-context concept learning exhibits a simplicity bias similar to one that has been observed in human concept learning. Section 3.5.1 outlines a possible strategy that is more computationally efficient to scale up future experiments. Despite the scale limitations in the experiments, the results in this thesis set the theoretical foundation for future research into learning biases of LLM in-context learning. Some directions for future work may include an in-depth error analysis on the LLM's concept learning behavior, extending the domain of concepts beyond numerical, and a detailed comparison between model and human behaviors. The main contribution of this thesis is showing that in-context learning (ICL) algorithms in LLMs can have similar learning biases as humans; ICL generally favors logical simplicity when presented with an unfamiliar concept. When two concepts have the same logical complexity (in our grammar), language models can still have drastically different performances on the corresponding concept learning tasks. The experiment on learning concepts ⟦more than⟧ and ⟦less than⟧ has demonstrated such learning biases. We argue that this bias is human-like, and unlike other forms of bias, the cause for this kind of bias seems more mysterious. For instance, biases such as race or gender in LLMs can potentially be attributed to texts about certain social groups in the corpus having more negative sentiments than positive (Park et al., 2021; Field et al., 2022). But for the bias in learning ⟦more than⟧ and ⟦less than⟧ concepts, it is not immediately clear what structure in the training data may enable this behavior. It can be an exciting direction to study this kind of phenomenon in more detail and investigate how such learning biases emerge in LLMs.

BIBLIOGRAPHY

Agmon, G., Loewenstein, Y., and Grodzinsky, Y. (2019). Measuring the cognitive cost of downward monotonicity by controlling for negative polarity. In *Glossa: a journal of general linguistics 4(1): 36.*

Agmon, G., Loewenstein, Y., and Grodzinsky, Y. (2022). Negative sentences exhibit a sustained effect in delayed verification tasks. *Journal of experimental psychology. Learning, memory, and cognition*, 48 1:122–141.

Akyürek, E., Wang, B., Kim, Y., and Andreas, J. (2024). In-context language learning: Architectures and algorithms. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F., editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 787–812. PMLR.

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. (2022). What learning algorithm is in-context learning? Investigations with linear models.

Aryabumi, V., Dang, J., Talupuru, D., Dash, S., Cairuz, D., Lin, H., Venkitesh, B., Smith, M., Marchisio, K., Ruder, S., Locatelli, A. F., Kreutzer, J., Frosst, N., Blunsom, P., Fadaee, M., Ustun, A., and Hooker, S. (2024). Aya 23: Open weight releases to further multilingual progress. *ArXiv*, abs/2405.15032.

Bach, K. and Fodor, J. A. (1998). Concepts: Where cognitive science went wrong.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler,

E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

Carcassi, F. and Szymanik, J. (2022). Neural networks track the logical complexity of boolean concepts. *Open Mind : Discoveries in Cognitive Science*, 6:132 – 146.

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

Feldman, J. (2000). Minimization of Boolean Complexity in Human Concept Learning. *Nature*, 407:630–633.

Field, A., Park, C. Y., and Tsvetkov, Y. (2022). Controlled analyses of social biases in wikipedia bios. *Proceedings of the ACM Web Conference 2022*.

Fodor, J. A. (1975). *The Language of Thought*. Harvard University Press.

Goodman, N. D., Tenenbaum, J. B., Feldman, J., and Griffiths, T. L. (2008). A rational analysis of rule-based concept learning. *Cognitive Science*, 32(1):108–154.

Goodman, N. D., Tenenbaum, J. B., and Gerstenberg, T. (2015). Concepts in a Probabilistic Language of Thought. In Margolis, E. and Laurence, S., editors, *The Conceptual Mind*, pages 623–654. The MIT Press.

Griffiths, T. L., Chater, N., and Tenenbaum, J. B. (2024). Bayesian models of cognition.

Liu, Z., Tan, B., Wang, H., Neiswanger, W., Tao, T., Li, H., Koto, F., Wang, Y., Sun, S., Pangarkar, O., Fan, R., Gu, Y., Miller, V., Ma, L., Tang, L., Ranjan, N., Zhuang, Y., He, G., Wang, R., Deng, M., Algayres, R., Li, Y., Shen, Z., Nakov, P., and Xing, E. (2025). Llm360 k2: Building a 65b 360-open-source large language model from scratch.

Marinescu, I., McCoy, R. T., and Griffiths, T. L. (2024). Distilling symbolic priors for concept learning into neural networks.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. (2022). Rethinking the role of demonstrations: What makes in-context learning work? In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Misra, K., Rayz, J., and Ettinger, A. (2023). COMPS: Conceptual minimal pair sentences for testing robust property knowledge and its inheritance in pre-trained language models. In Vlachos, A. and Augenstein, I., editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2928–2949, Dubrovnik, Croatia. Association for Computational Linguistics.

Murphy, G. L. (2002). The big book of concepts.

Park, C., Yan, X., Field, A., and Tsvetkov, Y. (2021). Multilingual contextual affective analysis of lgbt people portrayals in wikipedia. *Proceedings of the International AAAI Conference on Web and Social Media*, 15:479–490.

Piantadosi, S. T., Tenenbaum, J. B., and Goodman, N. D. (2016). The logical primitives of thought: Empirical foundations for compositional cognitive models. *Psychological Review*, 123(4):392–424.

Quilty-Dunn, J., Porot, N., and Mandelbaum, E. (2022). The Best Game in Town: The Re-Emergence of the Language of Thought Hypothesis Across the Cognitive Sciences. pages 1–55.

Riviere, G. T. M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ram'e, A., Ferret, J., Liu, P., Tafti, P. D., Friesen, A., Casbon, M., Ramos, S., Kumar, R., Lan, C. L., Jerome, S., Tsitsulin, A., Vieillard, N., Stańczyk, P., Girgin, S., Momchev, N., Hoffman, M., Thakoor, S., Grill, J.-B., Neyshabur, B., Walton, A., Severyn, A., Parrish, A., Ahmad, A., Hutchison, A., Abdagic, A., Carl, A., Shen, A.,

Brock, A., Coenen, A., Laforge, A., Paterson, A., Bastian, B., Piot, B., Wu, B., Royal, B., Chen, C., Kumar, C., Perry, C., Welty, C. A., Choquette-Choo, C. A., Sinopalnikov, D., Weinberger, D., Vijaykumar, D., Rogozi'nska, D., Herbison, D., Bandy, E., Wang, E., Noland, E., Moreira, E., Senter, E., Eltyshev, E., Visin, F., Rasskin, G., Wei, G., Cameron, G., Martins, G., Hashemi, H., Klimczak-Pluci'nska, H., Batra, H., Dhand, H., Nardini, I., Mein, J., Zhou, J., Svensson, J., Stanway, J., Chan, J., Zhou, J., Carrasqueira, J., Iljazi, J., Becker, J., Fernandez, J., van Amersfoort, J. R., Gordon, J., Lipschultz, J., Newlan, J., Ji, J., Mohamed, K., Badola, K., Black, K., Millican, K., McDonell, K., Nguyen, K., Sodhia, K., Greene, K., Sjoesund, L. L., Usui, L., Sifre, L., Heuermann, L., Lago, L., McNealus, L., Soares, L. B., Kilpatrick, L., Dixon, L., Martins, L., Reid, M., Singh, M., Iverson, M., Gorner, M., Velloso, M., Wirth, M., Davidow, M., Miller, M., Rahtz, M., Watson, M., Risdal, M., Kazemi, M., Moynihan, M., Zhang, M., Kahng, M., Park, M., Rahman, M., Khatwani, M., Dao, N., Bardoliwalla, N., Devanathan, N., Dumai, N., Chauhan, N., Wahltinez, O., Botarda, P., Barnes, P., Barham, P., Michel, P., Jin, P., Georgiev, P., Culliton, P., Kuppala, P., Comanescu, R., Merhej, R., Jana, R., Rokni, R., Agarwal, R., Mullins, R., Saadat, S., Carthy, S. M., Perrin, S., Arnold, S., Krause, S., Dai, S., Garg, S., Sheth, S., Ronstrom, S., Chan, S., Jordan, T., Yu, T., Eccles, T., Hennigan, T., Kociský, T., Doshi, T., Jain, V., Yadav, V., Meshram, V., Dharmadhikari, V., Barkley, W., Wei, W., Ye, W., Han, W., Kwon, W., Xu, X., Shen, Z., Gong, Z., Wei, Z., Cotruta, V., Kirk, P., Rao, A., Giang, M., Peran, L., Warkentin, T. B., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Sculley, D., Banks, J., Dragan, A., Petrov, S., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Borgeaud, S., Fiedel, N., Joulin, A., Kenealy, K., Dadashi, R., and Andreev, A. (2024). Gemma 2: Improving open language models at a practical size. *ArXiv*, abs/2408.00118.

Steinert-Threlkeld, S. (2021). Quantifiers in natural language: Efficient communication and degrees of semantic universals. *Entropy*, 23(10).

Tan, I., Kugler-Etinger, N., and Grodzinsky, Y. (2023). Do two negatives make a positive?

language and logic in language processing. *Language, Cognition and Neuroscience*, 38:1027 – 1043.

van de Pol, I., Lodder, P., van Maanen, L., Steinert-Threlkeld, S., and Szymanik, J. (2022). Quantifiers satisfying semantic universals have shorter minimal description length. *Cognition*, 232.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wang, L. Z., McCoy, R. T., and Steinert-Threlkeld, S. (2024). Minimization of boolean complexity in in-context concept learning.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., Xu, J., Zhou, J., Bai, J., He, J., Lin, J., Dang, K., Lu, K., Chen, K.-Y., Yang, K., Li, M., Xue, M., Ni, N., Zhang, P., Wang, P., Peng, R., Men, R., Gao, R., Lin, R., Wang, S., Bai, S., Tan, S., Zhu, T., Li, T., Liu, T., Ge, W., Deng, X., Zhou, X., Ren, X., Zhang, X., Wei, X., Ren, X., Fan, Y., Yao, Y., Zhang, Y., Wan, Y., Chu, Y., Cui, Z., Zhang, Z., and Fan, Z.-W. (2024). Qwen2 technical report. *ArXiv*, abs/2407.10671.

Z. Wang, L. and Steinert-Threlkeld, S. (2023). GQG: Generalized quantifier generalization - a dataset for evaluating quantifier semantics understanding in language models. In Hupkes, D., Dankers, V., Batsuren, K., Sinha, K., Kazemnejad, A., Christodoulopoulos, C., Cotterell, R., and Bruni, E., editors, *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, pages 185–192, Singapore. Association for Computational Linguistics.