BLOOM LARGE LANGUAGE MODELS AND THE CHOMSKY HIERARCHY

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the performance of BLOOM large language models of different sizes on understanding 12 different languages from the Chomsky hierarchy using few-shot prompts. We investigate whether an increase in the complexity of the languages learned by the larger models be characterized using the Chomsky hierarchy. We first show that prompting in BLOOM models enables reasoning with a good accuracy on language tasks as diverse as stack manipulation, string reversal, occurrence comparison, odds first, and interlocked pairing, when the queries are over short strings, that is, small bitwidth bit-vectors from the language. Second, we discover that the two largest models have the highest accuracy on such tasks for prompts with a fixed length but smaller models are able to achieve similar accuracies with longer prompts. Unlike classical automata or grammar based approaches where algorithms for more complex languages in the Chomsky hierarchy can also recognize simpler languages, we find that the performance of the BLOOM large language models cannot be explained by the complexity of the languages in the Chomsky hierarchy.

1 INTRODUCTION

Generalized pre-training of transformers and other models on large multidomain multilanguage text corpora have demonstrated unparalleled performance on multiple different language processing tasks for a few years now. More recently, it has been shown that the fine-tuning step of retraining such models on thousands or tens of thousands of examples can be replaced merely by few-shot demonstrations or prompting. The accuracy of such prompts for large language models is strongly comparable to fine-tuning using additional data for many tasks (Brown et al., 2020b).

The Chomsky hierarchy (Martin, 2022; Chomsky, 1959) is a classical approach for formal classification of languages into different complexity classes: regular languages, deterministic/non-deterministic context-free and context-sensitive languages. Formal models of increasing complexity such as deterministic finite automata, deterministic pushdown automata, non-deterministic pushdown automata, and non-deterministic Turing machines with linear space can recognize each of these languages - thereby, establishing a hierarchy of increasing complexity among these languages. Automata and grammars suitable for languages in the higher echelons of the Chomsky hierarchy can always be used to reason about languages lower in the Chomsky hierarchy.

We investigate if prompting in BLOOM large language models respects the Chomsky hierarchy, that is, for a given model, does the accuracy decrease with increase in the complexity of the language? We employ the BLOOM model (BigScience, 2022) as multiple trained model weights for BLOOM are publicly available and the model may be deemed to be more open as it has been trained by multiple partners on a public compute infrastructure. The central contributions in this paper are:

1. We evaluate the efficacy of prompting for a large language model BLOOM-176B in understanding languages from different classes of the Chomsky hierarchy. We observe that prompting is capable of solving problems with a good accuracy from different echelons of the Chomsky hierarchy, such as stack manipulation, string reversal and interlocked pairing.
2. We analyze the influence of the length of prompts on the accuracy of BLOOM models on the different tasks from the Chomsky hierarchy. We find that an increase in the length of

our prompts generally leads to an increase in model accuracy, and the length of the needed prompt depends both on the model and the task.

3. We study the influence of the size of the BLOOM model on the accuracy of the different tasks from the Chomsky hierarchy. Based on our analysis of 5 different BLOOM models with sizes 0.56 billion, 1 billion, 3 billion, 7 billion and 176 billion parameters, we find that the two largest models have the highest accuracy on our tasks with a fixed length of the prompt but smaller models are able to achieve similar accuracies with longer prompts.
4. Unlike classical automata or other grammar-based approaches where algorithms for more complex languages in the Chomsky hierarchy can also recognize simpler languages, we find that the performance of the BLOOM large language models cannot be explained by the complexity of the languages in the Chomsky hierarchy. BLOOM models perform better on context-sensitive language tasks than simpler regular language tasks.

2 Summary of Results

Our analysis uses three languages at each of the four levels of the language hierarchy - Regular languages, Deterministic Context Free (DCF), Non-deterministic Context Free (NCF), and Context Sensitive (CS). We use the input strings in the language of different lengths ranging from 4 to 8 bits, and use BLOOM models of 5 different sizes. In Table 1, we present the accuracy of the BLOOM-146B model for all the languages for different input lengths, along with the accuracy of the random guess. Figure 1 shows how the improvement in the accuracy over random guess for the input length of 4 bits. We observe that the model is able to learn two context-sensitive languages very accurately. These models are trained using natural language which is known to be mildly context-sensitive (Shieber, 1985) which can explain its ability to learn languages with context, but the complexity of the languages learned by the larger models do not correspond to the Chomsky hierarchy and the model has very poor prediction accuracy on the much lower complexity regular languages. Even within the same level of Chomsky hierarchy, the model shows a significant variance in its learning accuracy. In Section 5, we analyze the accuracy across different model sizes, language complexity, input lengths, and prompt lengths.

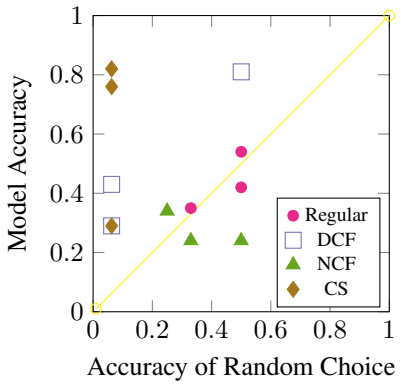


Figure 1: Prediction accuracy vs. random accuracy for languages from the Chomsky hierarchy for the BLOOM-176B model with input length 4.

Language Hierarchy	Problem	Random Accuracy	Accuracy (4 bits)	Accuracy (6 bits)	Accuracy (8 bits)
Regular Language	Parity Check	0.50	0.42	0.63	0.51
	Even Pairs	0.50	0.54	0.61	0.71
	Modular Arithmetic	0.33	0.35	0.37	0.24
Deterministic Context-Free Language	Reverse String	0.0625-0.004	0.29	0.11	0.10
	Stack Manipulation	0.0625-0.004	0.43	0.38	0.33
	Compare Occurrence	0.5	0.81	0.82	0.84
Non-deterministic Context-Free Language	Divide-by-2	0.25-0.125	0.34	0.25	0.27
	Equal Repeats	0.33	0.24	0.24	0.26
	Missing Palindrome	0.5	0.24	0.51	0.52
Context-Sensitive Language	Binary Addition	0.0625-0.004	0.82	0.21	0.03
	Odds First	0.0625-0.004	0.29	0.25	0.33
	Interlocked Pairing	0.0625-0.004	0.76	0.56	0.19

Table 1: BLOOM-146B accuracy on different language tasks in the Chomsky hierarchy.

3 RELATED WORK

Chomsky Hierarchy. With the goal of understanding the complexity of natural languages, the field of computational linguistics has sought to understand the place of natural languages within the hierarchy of formal languages. Since natural languages are considered to be "mildly" context-sensitive (Shieber, 1985; Jäger & Rogers, 2012), and context-sensitive languages generalize regular languages and context-free languages, it is plausible that large language models trained on natural language corpora can reason about languages at multiple levels of the Chomsky hierarchy. We investigate this hypothesis in our paper, and shown that prompting in BLOOM models can reason about language tasks with small bit-widths across multiple echelons in the Chomsky hierarchy. Formally, Chomsky's hierarchy is built on placing restrictions on the production rules $\phi \rightarrow \psi$ of a grammar. The following three restrictions create three layers of languages in the hierarchy:

1. Context-Sensitive: Consider the production rule $\phi \rightarrow \psi$, then there are ϕ_1, A, ϕ_2, and ω such that the production can be written as $\phi_1 A \phi_2 \rightarrow \phi_1 \omega \phi_2$. Hence, in general, the substitution of A with ω depends on the context ϕ_1 and ϕ_2.
2. Context-Free: Consider the production rule $\phi \rightarrow \psi$, then there are ϕ_1, A, ϕ_2, and ω such that the production can be written as $\phi_1 A \phi_2 \rightarrow \phi_1 \omega \phi_2$, and $A \rightarrow \omega$ i.e. ω is obtained from the symbol A without any knowledge of the context ϕ_1 and ϕ_2.
3. Regular: Consider the production rule $\phi \rightarrow \psi$, then there are $\phi_1, A, \phi_2, \omega, B$ and a such that the production can be written as $\phi_1 A \phi_2 \rightarrow \phi_1 \omega \phi_2$, $A \rightarrow \omega$, and $\omega = a$ or $\omega = Ba$. The last two restrictions are additional and allow the string to grow either with a non-terminal and a terminal or just a terminal symbol.

An extension of the Chomsky hierarchy is hypercomputation Siegelmann (2003; 2013), where the connection between hypercomputation and real-valued neural networks has been investigated. The connection between traditional neural networks and the Chomsky hierarchy has been studied (Delétang et al., 2022) with an emphasis on long short-term memory (LSTM), recurrent neural networks (RNNs) aided with deterministic stacks, non-deterministic stacks, and tapes, as well as relatively small transformers. Their goal was to train such neural networks enabled with appropriate stacks or tapes on languages from the Chomsky hierarchy. On the other hand, our goal is to understand if prompting in large language models respects the Chomsky hierarchy.

BLOOM Large Language Model. Transformers and their variants have been trained with autoregression or masking on large multi-language text corpora without an emphasis on a specific end task. Large language models such as BERT (Devlin et al., 2018), T5 (Raffel et al., 2020), GPT (Brown et al., 2020a), OPT (Zhang et al., 2022), PALM (Chowdhery et al., 2022) and BLOOM (BigScience, 2022) yield results close to the state-of-the-art on popular benchmarks in several language tasks. Subsequently, such generalized pre-trained transformers have achieved excellent performance on multiple tasks using fine-tuning, i.e., training over thousands of examples from the specific task. GPT-3, an autoregressive model, is a popular example of such a large language model that is available for use online or via an API after approval from the developers. BLOOM is an open-science open-access autoregressive multi-language model that is trained on the Jean Zay Public Supercomputer provided by the French government and is readily available to the public BigScience (2022). Unsupervised language modeling Jelinek & Mercer (1980) estimates the distribution of an input x from a set of examples $(x_1, x_2, \ldots, x_m)$ each composed of a sequence of symbols $(s_1, s_2, \ldots, s_n)$ by factorizing the joint distribution as the product of conditional distributions: $p(x) = \prod_{i=1}^{n} p(s_n \mid s_1, s_2, \ldots, s_{n-1})$. The need to compute such estimates using the Transformer framework Shoeybi et al. (2019) has led to the introduction of parallelism into both components of the transformer layer, the self-attention layer and the 2-layer multi-layer perceptron that follows it. A key insight has been to identify methods for parallelizing the General Matrix Multiply while minimizing the need for synchronization. BLOOM uses a corpus of 46 natural languages and 13 programming languages that correspond to 1.6 TB of pre-processed text or 350 billion unique tokens for training. The 70-layer 14336-dimensional model is a modified Megatron-LM GPT2 with a decoder-only architecture and employs ALiBI positional encodings with GeLU activation functions. We employ prompting on 5 pre-trained BLOOM models with an increasing number of parameters in our analysis. The simplest of these models has 560 million parameters while the most complex model has 176 billion parameters. We also employ intermediate BLOOM models with 1 billion, 3 billion and 7 billion parameters in our investigations.

4 PROMPTS FOR CHALLENGE LANGUAGE PROBLEMS

We prompt the BLOOM large language models (LLMs) on a benchmark of 12 languages (Dancette & Cord, 2022) drawn from different classes of the Chomsky hierarchy. Further, we can vary the length of the string from the language with which we prompt the model to increase the complexity of the task. Without loss of generality, we use bitvectors as strings and the bit width of the bitvectors provides another orthogonal dimension in which we scale a language task. This is in addition to the complexity of the language in the Chomsky hierarchy. Hence, we explore the capabilities of the BLOOM large language models across two different scaling dimensions. We describe the challenge language problems and the design of the prompts for these problems in this section. In our experiments, we also consider prompts of different lengths. We employ three different languages for each of the different types of language classes in the Chomsky hierarchy.

4.1 REGULAR LANGUAGES

For the class of regular languages, we consider the following problems: (i) Parity Check, (ii) Even Pairs, and (iii) Modular Addition. **Parity** recognizes if the input binary is of even parity, that is, $s \in (0^*10^*10^*)^*$. **Even Pairs** recognizes if the input binary has an even number of 01s and 10s, that is, $s \in ((00+11)^*(01+10)(00+11)^*(01+10)(00+11)^*)^*$. **Modular Addition** obtains the sum $((m+n) \mod 3)$ given two numbers m and n.

Parity: We created the following prompt to perform parity check on bit strings, that is, count if the number of 1s in the bit representation of an integer is odd or even. We use 20 examples in each prompt to guide the large language model BLOOM.

```
Sum of digits of 11100010 is 4; its parity is even.
Sum of digits of 10011101 is
```

The LLM predicts the following output in response to this query:

```
Sum of digits of 10011101 is 5; its parity is odd.
```

Even Pairs: We created prompts of the following form to detect an even number of 01 and 10 pairs in binary strings. The response to such natural language queries require reasoning over the basic facts. We use 3 examples in each prompt for 4 bits, and 20 examples for longer bit-widths.

```
The number of 01s in the string 10011001 is 2 and the number of 10s is 2, their sum is 4; hence, the class is 1.
The number of 01s in the string 11010011 is
```

The LLM predicts the following output in response to this query:

```
The number of 01s in the string 11011001 is 2 and the number of 10s is 2, their sum is 4; hence, the class is 1.
```

Modular Addition: We created the following prompt to perform modular addition on integers. We use 5 examples for addends with 4 bits, and 20 examples for long bit-widths.

```
The binary sum 1110 + 1001 in decimal is 14 + 923, and 23 % 3 = 2, which in binary is 10.
The binary sum 1000 + 1111 in decimal is
```

The LLM predicts the following output in response to this query:

```
The binary sum 1000 + 1111 in decimal is 12 + 1111, and 23 % 3 = 2, which in binary is 10.
```

4.2 DETERMINISTIC CONTEXT-FREE LANGUAGES

For the class of Deterministic Context Free Grammars, we employ the following problems: (i) String Reversal, (ii) Stack Manipulation and (iii) More 1s than 0s. **String Reversal** computes the reverse of a string. The string ww^r is produced by the following production rules of a context-free grammar: $S \to aSa, S \to bSb, S \to aa, S \to bb$. **Stack Manipulation** focuses on a stack with 0 and 1 entries and several push and pop operations; finally, we seek to identify the deterministic state of the stack after these operations. **More 1s than 0s** compares the number of 1s and 0s in a

string. A string with more number of 1s than 0s is given by the following context-free grammar: $S \rightarrow \epsilon|SS|1S0|0S1|1S|1$.

String Reversal: We created the following prompt to reverse a string. We use 20 examples in our prompts with 5 bits or more, and 5 examples for prompts with 4 bits.

```
Reverse of the binary string 11100010 is 01000111.
Reverse of the binary string 11000010 is
```

The LLM predicts the following output in response to this query:

```
Reverse of the binary string 11000010 is 01000011.
```

Stack Manipulation: We employ the following prompt to obtain the state of a stack after its manipulation. We use 5 examples in the prompt for final stacks with less than 5 items, and 20 examples in the prompt for deeper stacks.

```
Given the initial stack 10011101 and the operation sequence PUSH 1 POP POP POP PUSH 1 PUSH 1 POP POP the stack is 100111.
Given the initial stack 10110011 and the operation sequence PUSH 1 PUSH 1 POP POP POP POP PUSH 1 PUSH 1 the stack is
```

The LLM predicts the following output in response to this query:

```
Given the initial stack 10110011 and the operation sequence PUSH 1 PUSH 1 POP POP POP POP PUSH 1 PUSH 1 the stack is 10110011.
```

More 1s than 0s: We used the following prompt to determine if a string has more 1s than 0s. We use 3 examples for prompts with inputs of 4 bits or less, and 20 examples with inputs of longer bit-widths.

```
Number of 1s in 1101 is 3 and number of 0s in 1101 is 1; since 3>1, the class is 1.
Number of 1s in 1110 is
```

The LLM predicts the following output in response to this query:

```
Number of 1s in 1110 is 3 and number of 0s in 1110 is 2; since 3>2, the class is 1.
```

4.3 Non-deterministic Context-Free Languages

For the class of Non-deterministic Context Free Grammars, we employ the following problems: (i) Divide by 2, (ii) Equal Repeats, and (iii) Missing Palindrome. **Division by 2** produces a string $0^{\lceil n/2 \rceil}10^{\lfloor n/2 \rfloor + m}$ given an input string $0^{n-1}10^m$. **Equal Repeats** checks which fragment of a string has been repeated. Given a string of the form $0^l1^m0^n$ where $l = m$, $m = n$ or $l = n$, equal repeats identifies the string as class 0 if $l = m$; otherwise, it identifies as class 1 if $m = n$; otherwise, identifies as class 2 if $l = n$. Given a ternary string s with exactly one '2', **Missing Palindrome** determines if the '2' should be replaced by '1' or '0' to form a binary string that is a palindrome.

Division by 2: We design the following prompt to divide a number by 2 using its unary representation. We use 5 examples for 7 bits or more, 3 examples for 5 bits or more, and 2 examples for 4 bits. Changes to the prompt design such as explicitly comparing the number of zeroes and ones, or adding space between groups of zeros and ones do not lead to any substantial improvement despite the enhanced complexity.

```
Given the input 0000010, the number of 0s is 5 and 1 respectively, unary division by two produces 3 zeros and 4 zeros, that is 00010000.
Given the input 0001000,
```

The LLM predicts the following output in response to this query:

```
Given the input 0001000, the number of 0s is 3 and 3 respectively, unary division by two produces 2 zeros and 5 zeros, that is 00100000.
```

Equal Repeats: We employ the following prompt to identify the class of a string based on the equal

repeats in the string. We use 3 examples for repeating patterns with at most 4 bits and 5 examples for higher bit-widths.

The string 00111000 has 2 zeroes, 3 ones, and finally 3 zeros; hence the class is 1.
The string 000111000

The LLM predicts the following output in response to this query:

The string 000111000 has first 2 zeroes, then 3 ones, and finally 2 zeros; hence the class is 0.

Missing Palindrome: We use the following prompts to identify the missing character that causes a string to be a palindrome. We use 10 examples for prompts with 6 bits or more as using 20 examples causes an out-of-memory on 8 NVIDIA A100, and 5 examples for prompts with fewer bit-widths.

Replacing 2 by 1 in 1111 1211 yields 1111 1111, replacing 2 by 0 in 1111 1211 yields 1111 1011 ; since 1111 1111 is a palindrome, replace 2 by 1.
Replacing 2 by 1 in 1111 2111 yields

The LLM predicts the following output in response to this query:

Replacing 2 by 1 in 1111 2111 yields 1111 1111, replacing 2 by 0 in 1111 2111 yields 1111 0111 ; since 1111 1111 is a palindrome, replace 2 by 1.

4.4 Context Sensitive Languages

For the class of Context Sensitive Grammars, we employ the following problems: (i) Binary Addition, (ii) Interlocked Pairing, and (iii) Odds First. **Binary Addition** computes the sum $s_1 + s_2$ as a binary string given two binary strings s_1 and s_2. **Interlocked Pairs** creates the string $0^n1^{(m+n)}0^m$ given a string of the form 0^n1^m. **Odd-First** computes the strings composed of bits in odd and even positions respectively $s_1s_3s_5\ldots$ and $s_2s_4\ldots$ given a string $s_1s_2s_3\ldots s_n$.

Binary Addition: Since a prompt using straightforward binary addition produces low accuracies even for small bit widths, we employ the following prompt to perform binary addition on two strings. We use 20 examples for inputs with 5 or more bits, and 5 examples for inputs with 4 bits.

01001011 + 01101011 = 75 + 107 = 182 = 10110110.
01001000 + 01011011 =

The LLM predicts the following output in response to this query:

01001000 + 01011011 = 72 + 91 = 163 = 10100011.

Interlocked Pairs: We design the following prompt to create interlocked pairs $0^n1^{m+n}0^m$ from the input 0^n1^m. The use of the natural language mapping to integer addition enhances the accuracy of the model. In our experiments, we used 5 examples in the prompts for interlocked pairs for 7 or more bits, employed 3 examples for $5-6$ bits, and 2 examples for 4 bits.

Transform 000000 11 that is, 6 zeroes and 2 ones into 6 zeroes, 8 ones, 2 zeros, that is, 000000 11111111 00.
Transform 0000000 1 that is,

The LLM predicts the following output in response to this query:

Transform 0000000 1 that is, 7 zeroes and 1 ones into 7 zeroes, 8 ones, 1 zeros, that is, 0000000 11111111 0.

Odd-First: We use prompts of the following form to list the odd characters of a string before its even characters. We employ 20 examples for more than 5 bits, 10 examples for 5 bits, and 5 examples for 4 bits.

Given the string 11100010, odd followed by even characters are 1101 1000.
Given the string 11000010, odd followed by even characters are

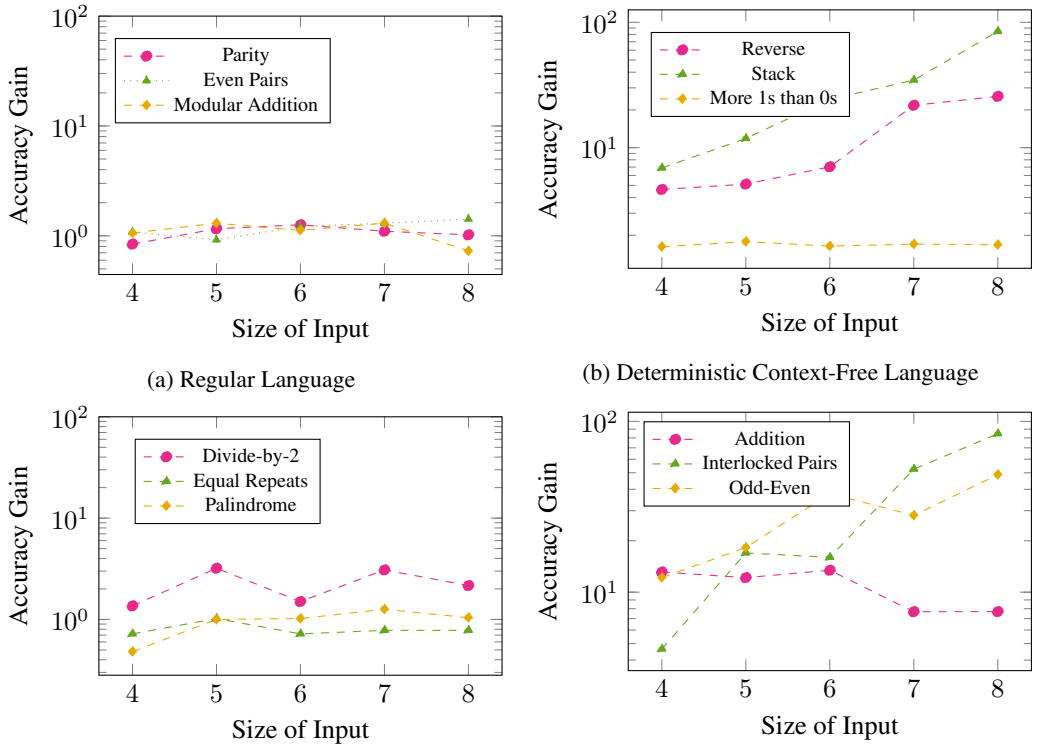
The LLM predicts the following output in response to this query:

Given the string 11000010, odd followed by even characters are 1001 1000.

5 RESULTS

We performed experiments on the BLOOM family of large language models on a server with 256 AMD cores, 2 TB of RAM and 8 Nvidia A100 80GB GPUs. Each language task was performed 100 times to compute the average accuracy of the task. Our analysis considers 4 dimensions: (i) the complexity of the language from the Chomsky hierarchy, (ii) the size of the input string from the language, (iii) the number of examples in the prompt, and (iv) the size of the model.

5.1 ACCURACY ON LANGUAGES IN THE CHOMSKY HIERARCHY



(a) Regular Language

(b) Deterministic Context-Free Language

(c) Nondeterministic Context-Free Language

(d) Context Sensitive Language

Figure 2: BLOOM-176B model shows high accuracy gain (prediction accuracy of the model / random guess probability) on the two deterministic context-free languages and context-sensitive languages. Note that the random guess probability is different for the different tasks and hence, the relative accuracy gain is a better evaluation metric. This increase in accuracy becomes more pronounced for these tasks as the input size increases. The model exhibits poor accuracy on the simpler regular languages. Thus, the Chomsky hierarchy exhibited by the grammar-based methods does not correspond to the languages learned by the large language model.

We evaluate the BLOOM-176B model on 3 language tasks each from increasingly complex classes from the Chomsky hierarchy: (i) regular language, (ii) deterministic context free language, (iii) nondeterministic context-free language, and (iv) context-sensitive language. The BLOOM-176B model is not effective at learning regular languages. Fig. 2a shows that modular arithmetic and parity detection do not perform better than random chance on 4 bits. BLOOM-176B perform slightly better than random chance on the problem of detecting if the number of 01s and 10s is even. Natural language is known to be mildly context-sensitive; so, our results indicate that BLOOM-176B trained mostly on natural languages finds examples of the regular language class challenging.

Prompting in the BLOOM-176B model works well for deterministic context-free languages as compared to random chance. Fig. 2b shows that prompting works well for the reversal of strings, stack manipulation, and detecting if a string has more number of 1s than 0s. This is different from automata and grammar models, where algorithms for deterministic context-free languages work well

for the contained subset of regular languages. Non-deterministic context-free languages perform close to random chance on the problems of equal repeat detection and missing palindrome digit determination. Prompting the model works better for Divide-by-2, as shown in Fig. 2c. Prompting the BLOOM-176B works well for context-sensitive languages shown in Fig. 2d. As natural languages are considered to be mildly context-sensitive, this suggests that BLOOM-176B trained on natural languages performs well on context-sensitive languages. The BLOOM large language model shows an interesting difference from traditional grammar and automata that generalize well from context-sensitive languages to less complex languages in the Chomsky hierarchy. This establishes the insufficiency of the existing language hierarchy to characterize the languages learned by increasingly larger language models and suggests the need for developing a new complexity hierarchy.

5.2 Influence of the Number of Examples in Prompts on Accuracy

The design of our prompts uses examples from the languages being investigated. A natural question to investigate is the influence of the number of examples in the prompts on the accuracy of the model. For this study, we use eight bitwidth examples from 3 languages across different layers of the Chomsky hierarchy, on which the model has a good baseline accuracy, and the accuracy has a potential room for improvement with more examples in the prompt. We analyze a small range 3-20 of number of examples on the large BLOOM-176B model as the use of larger number of examples leads to out-of-memory errors. Our results are shown in Fig. 3a. We investigate the use of a larger range 3-50 of examples on the relatively smaller BLOOM-7B model, which is still the largest of all the remaining BLOOM models. For the case of the missing palindrome task, we limited the number of examples as the system runs out of memory on a larger number of examples for BLOOM-176B, and we are constrained by the number of valid palindrome examples for BLOOM-7B. Fig. 3b shows the accuracy of the BLOOM-7B model as the length of the prompt is varied.

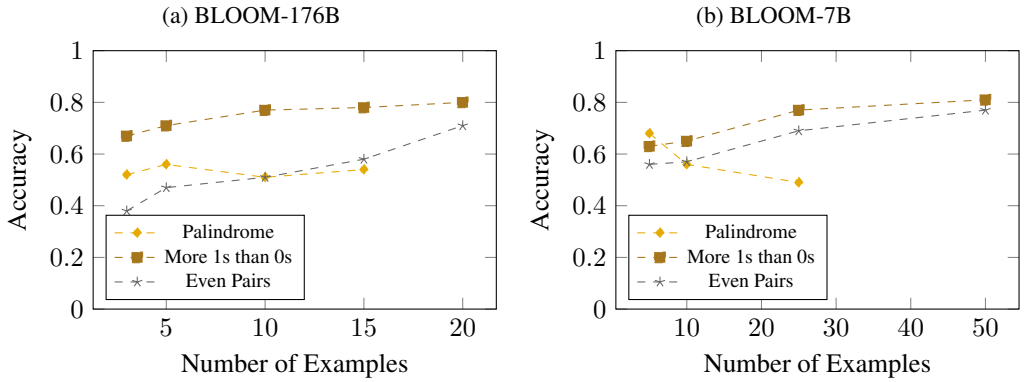
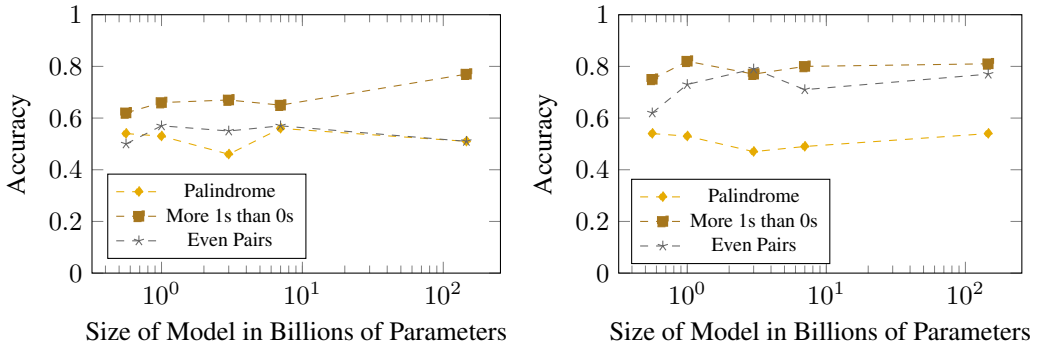


Figure 3: BLOOM-176B and BLOOM-7B accuracy with different number of examples in the prompts for the three languages.

We evaluate the accuracy of the model-prompt pairs by observing 100 independent and identically distributed samples drawn from a uniform random distribution. We conclude that an increase in the number of examples generally enhances the accuracy of the model-prompt pair.

5.3 Influence of Model Size on Accuracy

In our experiments, we also evaluate the change in accuracy over eight bitwidth examples drawn from different languages as the size of the BLOOM model increases. We employ prompting on 5 pre-trained BLOOM models. The smallest model has 560 million parameters, and the largest model has 176 billion parameters. The other three intermediate BLOOM models have 1 billion, 3 billion and 7 billion parameters. We use the same three languages from the different layers of the Chomsky hierarchy as used in Section 5.2 for this study. We consider two kinds of prompts for this study - the first fixes the number of examples from the language in the prompt to 10, and the second uses adaptive prompt sizes. The results from this study are illustrated in Figure 4. The accuracy of the model generally improves with an increase in the model size for fixed prompt sizes.



(a) 10 examples

(b) Adaptive Number of Examples

Figure 4: Accuracy of tasks using BLOOM models of different sizes - 560 million, 1 billion, 3 billion, 7 billion and, 176 billion parameters.

Fixed Prompt Length. In this case, we fix the prompt length to be 10. We observe that the prediction accuracy of the models generally increases with the number of parameters. This increase is not uniform across all languages. For the deterministic context-free language, More 1s than 0s, the improvement is significant. For the simpler regular language, Even Pairs, which is lower in the Chomsky hierarchy, even using the largest model does not yield a significant improvement.

Adaptive Prompt Length. We analyze the performance of the BLOOM models using adaptive prompt sizes. For each model, we adapt the number of examples to ensure that the 8 A100 GPU system is capable of generating a single response within 5 minutes without running out of memory. For the BLOOM-175B model, we use 20 examples except for the palindrome case (15 examples) due to memory issues. For the BLOOM-7B and other smaller models, we use 50 examples except for the palindrome case (25 examples) due to the number of available valid examples. Our adaptive prompting technique has better accuracy than the fixed-length prompt for smaller BLOOM models.

6 Conclusions and Future Work

There is a rich historical body of work in artificial intelligence connecting formal grammars to natural languages (Pullum & Gazdar, 1982; Tomita, 1985; Savitch, 1987; Kudlek et al., 2003). Natural languages have been considered as "mildly" context-sensitive and have been parsed by variants of context-free grammars (Tomita, 1984; Briscoe & Carroll, 1993). In this paper, we have explored this connection between natural languages and formal grammars. We evaluate the efficacy of prompting for large language BLOOM models in understanding languages from different classes of the Chomsky hierarchy Chomsky (1959) and observe that prompting is capable of solving problems with a good accuracy from different echelons of the Chomsky hierarchy, such as stack manipulation, string reversal and interlocked pairing. Unlike classical automata or other grammar-based approaches where algorithms for more complex languages in the Chomsky hierarchy can also recognize simpler languages, we find that the performance of the BLOOM large language models cannot be explained by the complexity of the languages in the Chomsky hierarchy. For example, BLOOM models perform better on context-sensitive language tasks than simpler regular language tasks. We also observe that increasing the size of the model or increasing the length of the prompts by including more examples from the language improves the prediction accuracy. We find that the two largest models have the highest accuracy on our tasks with a fixed length of the prompt, but smaller models are able to achieve similar accuracies with longer prompts.

Ethics Statement: Our approach brings to light a potential concern with BLOOM and possibly other large language models. The use of machine learning models in high-assurance applications necessitates a sound understanding of their generalization capability. While the larger language models exhibit good empirical prediction accuracy on many language tasks, our results show that the languages learned by these models do not conform to the traditional Chomsky hierarchy. Thus, there is a pressing need to develop a new hierarchy of concepts learned by the language models that can enable more accurate formal characterization of their learning capability.

REFERENCES

BigScience. BigScience language open-science open-access multilingual (BLOOM) language model. Online, May 2022.

Ted Briscoe and John A Carroll. Generalized probabilistic lr parsing of natural language (corpora) with unification-based grammars. *Computational linguistics*, 19(1):25–59, 1993.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and Others. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.*, 33:1877–1901, 2020a.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020b.

Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959.

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

Corentin Dancette and Matthieu Cord. Dynamic query selection for fast visual perceiver. May 2022.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Marcus Hutter, Shane Legg, and Pedro A Ortega. Neural networks and the chomsky hierarchy. July 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. October 2018.

Gerhard Jäger and James Rogers. Formal language theory: refining the chomsky hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1598):1956–1970, 2012.

Frederick Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *In Proceedings of the Workshop on Pattern Recognition in Practice*, pp. 381–397, Amsterdam, The Netherlands: North-Holland, May 1980.

Manfred Kudlek, Carlos Martín-Vide, Alexandru Mateescu, and Victor Mitrana. Contexts and the concept of mild context-sensitivity. *Linguistics and Philosophy*, 26(6):703–725, 2003.

John C Martin. *Introduction to Languages and the Theory of Computation*. 2022.

Geoffrey K Pullum and Gerald Gazdar. Natural languages and context-free languages. *Linguistics and Philosophy*, 4(4):471–504, 1982.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, and Others. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Walter J Savitch. Context-sensitive grammar and natural language syntax. In *The Formal complexity of natural language*, pp. 358–368. Springer, 1987.

Stuart M Shieber. Evidence against the context-freeness of natural language. In *Philosophy, language, and artificial intelligence*, pp. 79–89. Springer, 1985.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Hava T Siegelmann. Neural and super-turing computing. *Minds Mach.*, 13(1):103–114, 2003.

Hava T Siegelmann. Turing on Super-Turing and adaptivity. *Prog. Biophys. Mol. Biol.*, 113(1): 117–126, September 2013.

Masaru Tomita. Lr parsers for natural languages. In *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, pp. 354–357, 1984.

Masaru Tomita. An efficient context-free parsing algorithm for natural languages. In *IJCAI*, volume 2, pp. 756–764. Citeseer, 1985.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open pre-trained transformer language models. May 2022.