Targeted Syntactic Evaluation on the Chomsky Hierarchy

Taiga Someya, Ryo Yoshida, Yohei Oseki
The University of Tokyo
{taiga98-0809, yoshiryo0617, oseki}@g.ecc.u-tokyo.ac.jp

Abstract

In this paper, we propose a novel evaluation paradigm for Targeted Syntactic Evaluations, where we assess how well language models can recognize linguistic phenomena situated at different levels of the Chomsky hierarchy. Specifically, we create formal languages that abstract four syntactic phenomena in natural languages, each identified at a different level of the Chomsky hierarchy, and use these to evaluate the capabilities of language models: (1) $(\text{Adj})^n$ NP, (2) NP^n VP^n, (3) Nested Dependency, and (4) Cross Serial Dependency. We first train three different language models (LSTM, Transformer LM, and Stack-RNN) on language modeling tasks and then evaluate them using pairs of a positive and a negative sentence by investigating whether they can assign a higher probability to the positive sentence than the negative one. Our result demonstrated that all language models have the ability to capture the structural patterns of the $(\text{Adj})^n$ NP formal language. However, LSTM and Transformer LM failed to capture NP^n VP^n language and no architectures can recognize nested dependency and Cross Serial dependency correctly. Neural language models, especially Transformer LMs, have exhibited high performance across a multitude of downstream tasks, leading to the perception that they possess an understanding of natural languages. However, our findings suggest that these models may not necessarily comprehend the syntactic structures that underlie natural language phenomena such as dependency. Rather, it appears that they may extend grammatical rules equivalent to regular grammars to approximate the rules governing dependencies.

Keywords: linguistics, targeted syntactic evaluation, formal language

1. Introduction

In the field of natural language processing (NLP), neural language models have achieved remarkable success in various downstream tasks (Wang et al., 2019b,c). To find out the underpinnings behind their success, the literature on Targeted Syntactic Evaluations (Linzen et al., 2016; Marvin and Linzen, 2018) has investigated what kind of linguistic phenomena these neural language models can and cannot capture.[1] Targeted Syntactic Evaluations first focused on subject-verb agreement in English and other European languages (Linzen et al., 2016; An et al., 2019; Mueller et al., 2020), and recently a lot of following-up work was done to cover a wide range of linguistic phenomena across languages (Wilcox et al., 2018; Gulordava et al., 2018; Ravfogel et al., 2018; Marvin and Linzen, 2018; Kann et al., 2019; Chowdhury and Zamparelli, 2018; Futrell et al., 2019; Warstadt et al., 2019; Da Costa and Chaves, 2020; Chaves, 2020; Mueller et al., 2020; Trotta et al., 2021; Xiang et al., 2021; Mikhailov et al., 2021).

This line of research in Targeted Syntactic Evaluations has enabled us to measure the performance of various language models on individual phenomena such as subject-verb agreement and ellipsis. However, much remains unknown about why these models excel in recognizing certain linguistic phenomena while struggling with others. In parallel, the field of formal language theory has made strides in understanding the properties and complexities of various languages, and some syntactic phenomena have been theoretically situated within the Chomsky hierarchy (Chomsky, 1957; Langendoen, 1981; Shieber, 1985).

Building upon these observations, our study proposes a novel evaluation paradigm that goes beyond measuring the performance on individual syntactic phenomena. We instead focus on assessing how well these models can recognize phenomena situated at different levels of the Chomsky hierarchy. Specifically, we have constructed formal languages that abstract four syntactic phenomena in natural languages, each identified at a different level of the Chomsky hierarchy.

In this paper, we focus on four syntactic phenomena: (1) $(\text{Adj})^n$ NP, which imitates the repetition of adjectives before nouns, (2) NP^n VP^n, which imitates the embedded sentences without grammatical agreement between noun phrases and verb phrases, as seen in Japanese, (3) Nested Dependency, which imitates the embedded sentences with the grammatical agreement between noun phrases and verb phrases as seen in English, and (4) Cross Serial Dependency, which im-

[1] As an anonymous reviewer correctly pointed out, the concept of evaluating language models using specially crafted datasets (test suites) rather than text corpora was already proposed by Lehmann et al. (1996) On the other hand, the unified evaluation of language models focusing specifically on grammatical phenomena is considered to have started later, as exemplified by the work of Linzen et al. (2016)

LREC-COLING 2024, pages 15595–15605
20-25 May, 2024.

	Targeted Syntactic Evaluation (Linzen et al., 2016)	Neural Networks and the Chomsky Hierarchy (e.g., Delétang et al., 2023)	Ours
Targeting phenomena in natural languages / Evaluating as a language model	✓		✓
Aligned with the Chomsky Hierarchy		✓	✓

Table 1: Comparison of existing evaluation methods with our propoed approach. Our study bridges the existing two evaluation methods by incorporating the perspective of where each syntactic phenomenon resides in the Chomsky hierarchy into the framework of Targeted Syntactic Evaluations (TSE).

itates the sentences with multiple dependencies crossing each other, as seen in Swiss German. For each formal language, we create five variants with varying vocabulary sizes and test three different language models (LSTM, Transformer LM, and Stack-RNN) against these formal languages in order to investigate their ability to recognize formal languages with different complexities.

Our experimental result demonstrated that all language models have the ability to capture the structural patterns of the $(\text{Adj})^n$ NP formal language. However, LSTM and Transformer LM failed to capture the NP^n VP^n formal language and no architectures can recognize the Nested Dependency and Cross Serial Dependency correctly. In recent years, neural language models, especially Transformer LMs, have exhibited a high performance across a multitude of NLP downstream tasks, leading to the perception that they possess an understanding of natural languages. However, our findings suggest that these models may not necessarily comprehend the syntactic structures that underlie natural language phenomena such as dependency. Rather, it appears that they may extend grammatical rules equivalent to regular grammars to approximate the rules governing dependencies.[2]

2. Related Work

The evaluation of language models has primarily been based on metrics such as perplexity, which provides an objective measure of their performance but lacks insight into their performance on specific downstream tasks. While large-scale benchmarks such as GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019a) provide valuable information in this regard, many recent studies have attempted to demonstrate that language models have learned the syntax of natural languages. One such study was conducted by Linzen et al. (2016), who used minimal pairs to investigate the sensitivity of language models to the subject-verb agreement in English. The results showed that LSTM language models are fairly sensitive to English subject-verb agreement. However, this study and other related studies (e.g., Marvin and Linzen, 2018; Futrell et al., 2019; Gulordava et al., 2018) only focused on a limited range of linguistic phenomena.

In order to tackle this problem, more recent studies have introduced large-scale datasets for comprehensive syntactic evaluations (Warstadt et al., 2019, 2020). These datasets have made it possible to target a wide range of linguistic phenomena, not just subject-verb agreement, and to more thoroughly analyze the linguistic performance of language models. Additionally, these datasets have been constructed for several languages besides English (Trotta et al., 2021; Xiang et al., 2021; Mikhailov et al., 2021), allowing us to verify if the results obtained in English also hold for other languages, and to make comparisons between languages. More recently, Hu et al. (2020) proposed a more sophisticated testing paradigm inspired by psycholinguistics, where they used a 2×2 paradigm with a strict, multi-fold success criterion.

However, a common limitation in these studies is that it is not always straightforward to discern the root causes when there are discrepancies in accuracy across different linguistic phenomena. In most cases, the analyses remain descriptive, merely cataloging the extent to which models perform well on various syntactic tasks. In this paper, we propose a new evaluation paradigm where we evaluate language models using formal languages that mimic actual syntactic, each identified at a different level of the Chomsky hierarchy. This allows for a more interpretable evaluation of various language models, addressing the shortcomings of previous methodologies.[3]

Another line of research has focused on the ability of neural language models to recognize formal languages (Bodón and Wiles, 2000; Bo-

[2] Our code is publicly available at https://github.com/osekilab/TSE-Chomsky

[3] Several studies have pointed out that using natural languages for evaluation poses the risk that language models may rely on superficial lexical cues to solve tasks, without actually grasping the underlying rules of various syntactic phenomena (e.g., Gulordava et al., 2018; Maudslay and Cotterell, 2021). The use of formal languages is also motivated by this potential problem.

den and Wiles, 2002; Suzgun et al., 2019; Bhattamishra et al., 2020; Ebrahimi et al., 2020; Hahn, 2020; Delétang et al., 2023). These studies have investigated, both theoretically and experimentally, how far neural language models can recognize formal languages in the Chomsky hierarchy. For example, RNNs and LSTMs can recognize some context-free languages (Rodriguez and Wiles, 1997; Skachkova et al., 2018; Suzgun et al., 2019) and some simple context-dependent languages (Bodón and Wiles, 2000; Boden and Wiles, 2002), while Transformers are not well positioned in the Chomsky hierarchy (Bhattamishra et al., 2020; Hahn, 2020).

However, existing studies do not necessarily train various architectures explicitly "as language models". Additionally, they focus solely on the sequence of non-terminal symbols that define the structure of the target languages, neglecting the types of terminal tokens within those non-terminals. As a result, these studies lack the consideration for a vocabulary with the richness and diversity found in natural language. Therefore, while our work shares a common ground with these studies in trying to identify language models' positions on the Chomsky Hierarchy and utilizing formal language for evaluating them, these studies and ours differ in crucial ways: these previous studies are primarily concerned with the generalization capabilities of neural networks in general, whereas our research specifically aims to understand the extent to which neural "language models" can capture complex phenomena inherent in "natural language" (Table 1). In this paper, we train each model under language modeling tasks, and adopt the evaluation paradigms used in existing studies on Targeted Syntactic Evaluation, but extend them by incorporating formal languages that mimic actual syntactic phenomena.

3. Experiments

3.1. Formal languages

In this subsection, we introduce the four types of formal languages investigated in this paper: $(\text{Adj})^n$ NP, NP^nVP^n, Nested Dependency, and Cross Serial Dependency.[4]

Formal definition of formal languages We define the formal languages investigated in this paper as follows: Let $V_{\text{non.}}$ be a set of finite non-terminal symbols. For each non-terminal symbol $A \in V_{\text{non.}}$, we define T terminal symbols $a_{A,0}, \cdots, a_{A,T}$. Next, we determine a set of finite-length strings consisting only of non-terminal symbols $L_{\text{non.}} \subseteq V^*_{\text{non.}}$. Then, each string contained in $L_{\text{non.}}$ is rewritten by replacing every non-terminal symbol A in it with one of $a_{A,0}, \cdots, a_{A,T}$. If a non-terminal symbol appears multiple times in a string, it may be replaced with different terminal symbols. The resulting set of strings consisting only of terminal symbols is defined as the language L. The language L is determined by the set of non-terminal symbols $V_{\text{non.}}$, the number T of terminal symbols corresponding to each non-terminal symbol, and the set $L_{\text{non.}}$ of non-terminal symbol strings. Note that $L_{\text{non.}}$ and L belong to the same class in the Chomsky hierarchy: If there is a grammar that generates $L_{\text{non.}}$, we can construct a grammar that generates language L by applying the rewriting rules $A \to a_{A,0}| \cdots |a_{A,T}$ to all the non-terminal symbols A. Conversely, if there is a grammar that generates language L, we can construct a grammar that generates $L_{\text{non.}}$ by applying the rewriting rules $a_{A,0} \to A, \cdots, a_{A,T} \to A$.

$(\text{Adj})^n$ NP The formal language $L_{\text{non.}}$ which belongs to this type is defined as follows:

$$V_{\text{non.}} = \{\text{Adj}, \text{NP}\} \quad (1)$$

$$L_{\text{non.}} = \{\text{Adj}^n\text{NP} : n > 0\} \quad (2)$$

This type of formal language corresponds to a linguistic phenomenon we observe in English: we can repeat an infinite number of adjectives before a noun. In (i), for example, we can infinitely repeat old before man and the sentence is still grammatical (Chomsky, 1957).

(i) The $(\text{old})^n$ man comes.

This corresponds to $V_{non.}$ with the following rewriting rules:

$$\text{Adj} \to \text{old} \quad (3)$$

$$\text{NP} \to \text{man} \quad (4)$$

This formal language is a regular language that can be easily recognized by a finite state automaton with two states: one for a sequence of Adj's and the other for the single occurrence of NP after the sequence of Adj's.

NP^n VP^n The formal language $L_{\text{non.}}$ which belongs to this type is defined as follows:

$$V_{\text{non.}} = \{\text{NP}, \text{VP}\} \quad (5)$$

$$L_{\text{non.}} = \{\text{NP}^n\text{VP}^n : n > 0\} \quad (6)$$

This type of formal language corresponds to a linguistic phenomenon we observe in Japanese. In (ii), for example, there are three NP (noun phrases) followed by three VP (verb phrases). There is no particular requirement for grammatical agreement between each NP and VP.

[4] Adj, NP, and VP indicate adjective, noun phrase, and verb phrase, respectively.

(ii) Taroo-ga Hanako-ga Ziroo-ga
Taroo-Nom Hanako-Nom Ziroo-Nom
hashitta to itta to omotta.
ran that said that thought.
'Taroo thought that Hanako said that Ziroo ran.'

This corresponds to $V_{non.}$ with the following rewriting rules:

$$\text{NP} \rightarrow \text{Taroo}|\text{Hanako}|\text{Ziroo}|\cdots \quad (7)$$
$$\text{VP} \rightarrow \text{hasitta}|\text{itta}|\text{omotta}|\cdots \quad (8)$$

This formal language cannot be generated by a finite state automaton, since it requires memory to keep track of the number of NP's generated so far, thus it is not a regular language. However, it is a context-free language (Chomsky, 1957), which is one level higher in the Chomsky hierarchy.

Nested Dependency The formal language $L_{\text{non.}}$ which belongs to this type is defined as follows:

$$V_{\text{non.}} = \{\text{NP}_0, \cdots, \text{NP}_{N-1}, \text{VP}_0, \cdots, \text{VP}_{N-1}\} \quad (9)$$
$$L_{\text{non.}} = \{\text{NP}_{i_0} \cdots \text{NP}_{i_{n-1}}, \text{VP}_{i_{n-1}} \cdots \text{VP}_{i_0} : n \geq 0;\ 0 \leq\ i_0, \cdots, i_{n-1} \leq N-1\} \quad (10)$$

This type of formal language corresponds to a linguistic phenomenon we observe in English. In (iii), for example, the plural noun phrase the dogs must agree in number with the plural verb bark, and the singular noun phrase the cat with the singular verb chases.

(iii) The dogs that the cat chases bark.

This corresponds to $V_{non.}$ with the following non-terminal symbols and rewriting rules:

$$V_{\text{non.}} = \{\text{NP}_{\text{singular}}, \text{NP}_{\text{plural}} \cdots, \text{VP}_{\text{singular}}, \text{VP}_{\text{plural}}, \cdots\} \quad (11)$$
$$\text{NP}_{\text{singular}} \rightarrow \text{the cat}|\text{the dog}|\cdots \quad (12)$$
$$\text{NP}_{\text{plural}} \rightarrow \text{the cats}|\text{the dogs}|\cdots \quad (13)$$
$$\text{VP}_{\text{singular}} \rightarrow \text{chases}|\text{barks}|\cdots \quad (14)$$
$$\text{VP}_{\text{plural}} \rightarrow \text{chase}|\text{bark}|\cdots \quad (15)$$

This formal language is also known as a context-free language (Chomsky, 1957), which is one level higher in the Chomsky hierarchy than a regular language.

Cross Serial Dependency The formal language $L_{\text{non.}}$ which belongs to this type is defined as follows:

$$V_{\text{non.}} = \{\text{NP}_0, \cdots, \text{NP}_{N-1}, \text{VP}_0, \cdots, \text{VP}_{N-1}\} \quad (16)$$
$$L_{\text{non.}} = \{\text{NP}_{i_0} \cdots \text{NP}_{i_{n-1}} \text{VP}_{i_0} \cdots \text{VP}_{i_{n-1}} : n \geq 0;\ 0 \leq\ i_0, \cdots, i_{n-1} \leq N-1\} \quad (17)$$

This type of formal language corresponds to a linguistic phenomenon we observe in Swiss German. In (iv), for example, the dative verb hälfe has the dative noun phrase em Hans as its argument, and the accusative verb aastriiche has the accusative noun phrase es huus as its argument. Here, there is a syntactic dependency between the verb and its argument: they should have the same case-marking (Shieber, 1985).

(iv) ... mer em Hans es huus hälfed
... we Hans-DAT the house-ACC helped
aastriiche.
paint.
'... we helped Hans paint the hause.'

This corresponds to $V_{non.}$ with the non-terminal symbols and the rewriting rules:

$$V_{\text{non.}} = \{\text{NP}_{\text{dative}}, \text{NP}_{\text{accusative}} \cdots \text{VP}_{\text{dative}}, \text{VP}_{\text{accusative}}, \cdots\} \quad (18)$$
$$\text{NP}_{\text{dative}} \rightarrow \text{em Hans}|\text{em huus}|\cdots \quad (19)$$
$$\text{NP}_{\text{accusative}} \rightarrow \text{de Hans}|\text{es huus}|\cdots \quad (20)$$
$$\text{VP}_{\text{dative}} \rightarrow \text{hälfe}|\cdots \quad (21)$$
$$\text{VP}_{\text{accusative}} \rightarrow \text{aastriiche}|\cdots \quad (22)$$

This formal language is known as a context-dependent language, also called a copy language (Aho and Ullman, 1972), which is one level higher in the Chomsky hierarchy than a context-free language.

3.2. Data Generation

In this paper, we perform a BLiMP-style Targeted Syntactic Evaluation (Warstadt et al., 2020): we first train a language model on a language modeling task on positive examples and then evaluate the language model by investigating whether the language model can assign a higher probability to positive examples than negative ones. We prepare data for each formal language defined in the previous subsection with five different numbers of terminal symbols (= T in section 3.1): 5, 10, 50, 100, and 500. This results in a total of 24 formal languages. For simplicity, we only test the Nested Dependency and Cross Serial Dependencys with three non-terminal symbols ($V_{\text{non.}} = \{\text{NP}_0, \cdots, \text{NP}_2, \text{VP}_0, \cdots, \text{VP}_2\}$).

3.2.1. Data size and data split

For each formal language, we create 110,000 random samples of strings of length $l \sim \mathcal{U}(4, 30)$. We use 90,000 samples as train data, and 10,000 and 10,000 samples for validation and "in-dist" test data, respectively. We also create "out-of-dist" test data by sampling 10,000 strings of length $l \sim \mathcal{U}(31, 100)$ from each grammar, to evaluate

Data type	Positive example	Negative examples
$(\text{Adj})^n$ NP (Regular)	Adj Adj Adj Adj Adj NP	Adj Adj Adj **NP** Adj NP Adj **NP NP NP** Adj NP
NP^n VP^n (Context-free)	NP NP NP VP VP VP	NP NP VP VP VP NP NP NP VP VP VP **VP**
Nested Dependency (Context-free)	NP_1 NP_3 NP_4 VP_4 VP_3 VP_1	NP_1 NP_3 NP_4 **VP_1** VP_3 **VP_4** NP_1 **NP_4** **NP_3** VP_4 VP_3 VP_1
Cross Serial Dependency (Context-sensitive)	NP_3 NP_2 NP_1 VP_3 VP_2 VP_1	NP_3 NP_2 NP_1 **VP_2** **VP_3** VP_1 NP_3 **NP_1** **NP_2** VP_3 VP_2 VP_1

Table 2: Examples of four types of formal languages investigated in this paper and the positions on the Chomsky Hierarchy. Here, we demonstrate patterns only with non-terminal symbols and ignore terminal symbols.

whether the language model can generalize to longer strings than those seen during training. Finally, we generate negative examples for "in-dist" and "out-of-dist" test data in the manner described in Subsection 3.2.3, to create minimal pairs for evaluation.

3.2.2. Generating positive examples

We generate positive examples for each formal language by first sampling the necessary number of non-terminal symbol sequences belonging to the formal language and then replacing each non-terminal symbol with a terminal symbol based on the rewriting rules (Algorithm 1). Here, each terminal symbol is selected from the set of terminal symbols that can be rewritten from the target non-terminal symbol according to the normal distribution.

Algorithm 1 Algorithm for Generating Positive Examples from a Given Formal Language

function GENERATE SAMPLE($L_{\text{non.}}$)
  $S_{\text{non.}} \leftarrow$ a sequence of non-terminals sampled from $L_{\text{non.}}$
  $string \leftarrow \emptyset$
  for $i \leftarrow 0, length(S_{\text{non.}}) - 1$ **do**
    $s_{\text{non.}} \leftarrow S_{\text{non.}}[i]$
    $V \leftarrow \{v \mid s_{\text{non.}} \rightarrow v\}$ ▷ Rewriting rules
    $v \leftarrow v$ sampled from V according to Normal Distribution
    $string.append(v)$
  end for
  return $string$
end function

3.2.3. Generating negative examples

To prepare minimal pairs for evaluation, we generate negative examples for 'in-dist" and "out-of-dist" test data in the following way (cf. Table 2):

$(\text{Adj})^n$ NP We generate a negative example by replacing $k \sim \mathcal{U}(1, l-1)$ occurrences of Adj's with NP's in a corresponding positive example of length l.

NP^nVP^n We generate a negative example by either adding or removing a single occurrence of NP or VP in a corresponding positive example.

Nested/Cross Serial Dependency We generate a negative example by selecting two positions within the NP or VP and swapping their occurrences in a corresponding positive example. We create negative examples by breaking dependencies between non-terminals because we are interested in whether the language models can successfully capture the dependencies between non-terminals.

3.3. Models

In this paper, we evaluate the performance of the following three neural language models using our formal languages. In this study, we are particularly interested in understanding the performance differences across various architectures. Therefore, to ensure a fair comparison, we keep the number of layers and the dimensionality of embeddings consistent across all models under investigation.

LSTM A 2-layer LSTM (Hochreiter and Schmidhuber, 1997) language model with 256-dimensional word embedding and 256-dimensional hidden layer, implemented in PyTorch.[5] The total number of parameters is around 1.05M.

Transformer LM A 5-layer, 4-head Transformer (Vaswani et al., 2017) decoder with 128-dimensional word embedding, 512-dimensional

[5] https://pytorch.org/

Data Type	#Terms	LSTM		Stack-RNN		Transformer LM	
		in-dist	out-of-dist	in-dist	out-of-dist	in-dist	out-of-dist
(Adj)^n NP (Regular)	5	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**99.83**
	10	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**99.94**
	50	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**99.97**
	100	**99.82**	**99.99**	**100.0**	**100.0**	**100.0**	**99.96**
	500	**92.06**	**97.80**	**93.22**	**99.13**	**99.94**	**99.98**
NP^{n}VP^n (Context-free)	5	**99.94**	65.83	**100.0**	**100.0**	**99.76**	55.06
	10	**99.61**	62.12	**100.0**	**100.0**	**99.55**	57.47
	50	64.79	50.72	**99.97**	**100.0**	**97.56**	60.69
	100	53.31	50.89	**99.80**	**99.94**	**95.45**	61.07
	500	55.70	50.79	52.85	51.61	**94.63**	59.99
Nested Dependency (Context-free)	5	55.59	48.41	47.47	47.69	83.98	57.42
	10	49.27	49.07	49.71	49.35	84.18	56.74
	50	50.58	50.57	50.73	50.06	76.82	52.66
	100	50.99	50.70	50.76	51.04	51.50	50.71
	500	51.12	51.03	50.61	50.70	51.15	50.76
Cross Serial Dependency (Context-sensitive)	5	62.13	54.04	80.65	79.69	84.05	58.00
	10	53.49	49.71	81.71	81.44	83.62	57.28
	50	50.37	50.19	51.53	50.44	78.56	54.08
	100	50.58	50.79	50.57	50.73	51.04	51.14
	500	50.61	50.94	50.37	51.17	50.97	50.61

Table 3: The accuracy on the in-dist/out-of-dist test data for each formal language and architecture. For each architecture, we report the score for the model that exhibits the lowest perplexity on the validation set among the 30 models (10 random seeds $\times$ 3 learning rates). The values greater than 90% are shown in bold. In-dist test data consists of strings with the same length distribution as in train/dev data, while out-of-dist test data consists of longer strings than in train/dev data.

feedforward layer, and sinusoidal word encoding, implemented in PyTorch. The total number of parameters is around 991k.

Stack-RNN A single-layer RNN language model with 256-dimensional word embedding and 256-dimensional hidden layer, augmented with a differentiable stack of size 150 (Joulin and Mikolov, 2015). Each cell of the stack contains an 8-dimensional vector. The total number of parameters is around 136k.

Training and Evaluation Configurations We use SGD for LSTM models and AdamW (Loshchilov and Hutter, 2019) for Transformer LM and Stack-RNN model. Each language model is trained for 15 epochs with a batch size of 512, using 10 different random seeds and 3 different learning rates (0.1/0.2/0.3 for LSTM models, 0.0001/0.0003/0.0005 for Transformer LM and Stack-RNN model).[6] Each model is evaluated through the percentage of pairs where the language model successfully assigned a higher probability to the positive example than the negative one in "in-dist" and "out-of-dist" test data:[7]

$$P(S_{positive}) > P(S_{negative})$$

For each architecture under consideration, we report the score for the model that exhibits the lowest perplexity on the validation set among the 30 models (10 random seeds $\times$ 3 learning rates).

4. Results and Discussion

The results are presented in Table 2, which shows the scores for each architecture on in-dist/out-of-dist test data. We considered architectures that achieved an accuracy greater than 90% to have successfully captured the rules of each data.[8] In the following sections, we organize the results and provide our analysis for each formal language.

[6] All the language models are trained on NVIDIA V100 GPU with 16GB memory.

[7] In the case of NP^n VP^n, the lengths of positive and negative examples are different, so the probability of each example is calculated as the average probability assigned to each word contained in the sentence.

[8] This is because we are interested in whether these language models can recognize our formal languages at all, and this evaluation metric mostly follows that of Delétang et al. (2023)

(Adj)^n NP For this type of formal language, all architectures have successfully captured the rules for both in-dist and out-of-dist test data, regardless of the number of terminal symbols. This result bolsters and is consistent with the results reported in previous studies that LSTM and Stack-RNN can recognize regular languages (e.g., Delétang et al., 2023). The result is not necessarily consistent with the reported results that Transformer cannot recognize a part of regular languages (Bhattamishra et al., 2020; Delétang et al., 2023), but this may be due to the fact that (Adj)^n NP is among the simplest regular languages. Additionally, the result is also consistent with the results that LSTM and Transformer can successfully capture those local dependencies found in morphology or internal structure of noun phrases (Warstadt et al., 2020; Someya and Oseki, 2023). Furthermore, the high accuracy observed in this formal language suggests that each architecture can capture this syntactic phenomenon across languages without relying on superficial lexical cues found in natural languages (cf. Gulordava et al., 2018; Maudslay and Cotterell, 2021).

NP^n VP^n For in-dist test data, Transformer LM and Stack-RNN models were able to successfully capture the rules, while LSTM failed when the vocabulary size is large. However, only the Stack-RNN model was able to generalize correctly on out-of-dist data, where the model is tested with those sentences that are strictly longer than the sentences in train/dev data. This result is not consistent with the previous results that LSTM models can capture context-free language (e.g., Weiss et al., 2018; Wiles and Elman, 1995), and suggests that the results do not hold true when tested following targeted syntactic evaluation approach (e.g., Linzen et al., 2016): models are trained on a language modeling task and each non-terminal symbols in formal language have corresponding terminal symbols. Furthermore, it is suggested that novel architectures, such as those incorporating stack memory mechanisms, may be necessary to capture such phenomena.

Nested Dependency For in-dist test data, Transformer LM performs fairly well when the vocabulary size is small, while the other architectures failed in all the cases. As for out-of-dist test data, no architectures were able to generalize the rules correctly. This is not necessarily consistent with the previous results that Stack-RNN can recognize palindrome languages, and suggests that the results does not hold true when tested following targeted syntactic evaluation approach and that the model cannot recognize the corresponding syntactic phenomenon in natural languages, i.e., nested dependencies. Additionally, these results are not consistent with the previous findings that LSTM and Transformer-based language models can capture English center embedding to some extent (Wilcox et al., 2019; Hu et al., 2020), suggesting that these language models may not necessarily capture the rules of center embedding, but rather solve the task using lexical information such as co-occurrence or frequency.

Cross Serial Dependency First of all, this phenomenon has not been observed in languages that have been the subject of Targeted Syntactic Evaluation thus far (e.g., Linzen et al., 2016; Mueller et al., 2020). Our work is the first to evaluate this phenomenon using a formal language. In terms of the in-dist test data, Stack-RNN and Transformer LM perform fairly well when the number of terminal symbols was small, while LSTM failed in all the cases. As for the out-of-dist test data, Strack-RNN perform comparably well when the vocabulary size is small, while no architectures were able to generalize the rules correctly. These results suggest that novel architectures are required to capture this syntactic phenomenon.

Summary All language models demonstrated the ability to capture the structural patterns of the (Adj)^n NP formal language, even with increased vocabulary size. However, LSTM and Transformer LM failed to capture NP^n VP^n language and no architectures can recognize nested dependency and Cross Serial dependency correctly. Since their introduction, Transformer LMs have exhibited high performance across a multitude of downstream tasks (Wang et al., 2018, 2019a) leading to the perception that they possess an understanding of natural languages. However, our findings suggest that these models may not necessarily comprehend the syntactic structures that underlie natural language phenomena such as dependency. Rather, it appears that they may extend grammatical rules equivalent to Regular grammars to approximate the rules governing dependencies.

5. Conclusion

Existing research in the realm of Targeted Syntactic Evaluations has primarily focused on empirically assessing the capabilities of various language models to capture specific syntactic phenomena present in natural languages. However, these studies have largely remained observational, evaluating which syntactic phenomena a given language model can or cannot capture, without providing substantive insights into the reasons behind these results. Separately, another line of research has examined the extent to which different neural

models can recognize languages belonging to specific classes within the Chomsky hierarchy.

In this paper, we propose a novel evaluation paradigm for Targeted Syntactic Evaluations, where we assess how well language models can recognize linguistic phenomena situated at different levels of the Chomsky hierarchy. Specifically, we created formal languages that abstract four syntactic phenomena in natural languages, each identified at a different level of the Chomsky hierarchy, and used these to evaluate the capabilities of language models: (1) $(\text{Adj})^n$ NP, (2) NP^n VP^n, (3) Nested Dependency, and (4) Cross Serial Dependency. We first trained three different language models (LSTM, Transformer LM, and Stack-RNN) on language modeling tasks and then evaluated them using pairs of a positive and a negative sentence by investigating whether they can assign a higher probability to the positive sentence than the negative one. Our study bridges the two existing research paradigms by incorporating the perspective of where each syntactic phenomenon resides in the Chomsky hierarchy into the framework of Targeted Syntactic Evaluations. This allows us to scrutinize the capabilities of various language models in capturing syntactic phenomena in natural languages, in terms of their positioning within the Chomsky hierarchy. Moreover, by utilizing formal languages instead of natural languages for evaluation, we mitigate the risk of language models relying on superficial lexical cues rather than capturing the rules governing different syntactic phenomena.

Our experimental result demonstrated that all language models have the ability to capture the structural patterns of the $(\text{Adj})^n$ NP formal language. However, LSTM and Transformer LM failed to capture NP^n VP^n language and no architectures can recognize nested dependency and Cross Serial dependency correctly. Transformer LMs have exhibited high performance across a multitude of downstream tasks (Wang et al., 2018, 2019a) leading to the perception that they possess an understanding of natural languages. However, our findings suggest that these models may not necessarily comprehend the syntactic structures that underlie natural language phenomena such as dependency. Rather, it appears that they may extend grammatical rules equivalent to Regular grammars to approximate the rules governing dependencies.

Possible directions for future research are as follows. First, an immediate objective could be to broaden the range of syntactic phenomena studied, thereby enhancing the robustness and generalizability of our findings. Second, evaluating the performance of recently emerged pretrained language models on this specialized benchmark would offer important insights into their ability to capture the underlying syntactic rules, rather than merely relying on superficial lexical cues. Finally, the introduction of novel architectures that performs well in this benchmark could provide new perspectives on how to build models that genuinely understand language structures, rather than approximating understanding through lexical shortcuts.

Limitations

In this study, we only tested a limited variety of models. To evaluate whether each architecture was able to capture the rules of each formal language, we trained 30 models for each architecture using 10 different random seeds and 3 different learning rates. To fairly compare the performance of the architectures, we kept the number of parameters roughly the same for each architecture. However, there are other hyperparameters such as embedding dimensions and the number of layers that vary between architectures. Therefore, it is possible that some architectures may perform better with different hyperparameters.

Acknowledgements

This work was supported by JST PRESTO Grant Number JPMJPR21C2, Japan.

6. Bibliographical References

Alfred V Aho and Jeffrey D Ullman. 1972. *The Theory of Parsing, Translation and Compiling. Parsing, vol. I*. Prentice-Hall, Englewood Cliffs.

Aixiu An, Peng Qian, Ethan Wilcox, and Roger Levy. 2019. Representation of constituents in neural language models: Coordination phrase as a case study. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2888–2899, Hong Kong, China. Association for Computational Linguistics.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online. Association for Computational Linguistics.

Mikael Boden and Janet Wiles. 2002. On learning context free and context sensitive languages. *IEEE Transactions on Neural Networks*, 13:491–493.

Mikael Bodón and Janet Wiles. 2000. Context-free and context-sensitive dynamics in recurrent neural networks. *Connection Science*, 12(3-4):197–210.

Rui P Chaves. 2020. What don't RNN language models learn about Filler-Gap dependencies? *Proceedings of the Society for Computation in Linguistics*, 3(1):20–30.

Noam Chomsky. 1957. *Syntactic structures*. Mouton.

Shammur Absar Chowdhury and Roberto Zamparelli. 2018. RNN simulations of grammaticality judgments on long-distance dependencies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 133–144, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jillian Da Costa and Rui Chaves. 2020. Assessing the ability of transformer-based neural models to represent structurally unbounded dependencies. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 12–21, New York, New York. Association for Computational Linguistics.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A Ortega. 2023. Neural networks and the chomsky hierarchy. In *The Eleventh International Conference on Learning Representations*.

Javid Ebrahimi, Dhruv Gelda, and Wei Zhang. 2020. How can self-attention networks recognize dyck-n languages? *CoRR*, abs/2010.04303.

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. 2019. Neural language models as psycholinguistic subjects: Representations of syntactic state. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 32–42, Minneapolis, Minnesota. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

S Hochreiter and J Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the Association of Computational Linguistics*.

Armand Joulin and Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 190â 198, Cambridge, MA, USA. MIT Press.

Katharina Kann, Alex Warstadt, Adina Williams, and Samuel R Bowman. 2019. Verb argument structure alternations in word and sentence embeddings. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 287–297.

D. Terence Langendoen. 1981. The generative capacity of word-formation components. *Linguistic Inquiry*, 12(2):320–322.

Sabine Lehmann, Stephan Oepen, Sylvie Regnier-Prost, Klaus Netter, Veronika Lux, Judith Klein, Kirsten Falkedal, Frederik Fouvry, Dominique Estival, Eva Dauphin, Herve Compagnion, Judith Baur, Lorna Balkan, and Doug Arnold. 1996. TSNLP - test suites for natural language processing. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn Syntax-Sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Rowan Hall Maudslay and Ryan Cotterell. 2021. Do syntactic probes probe syntax? experiments with jabberwocky probing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 124–131, Online. Association for Computational Linguistics.

Vladislav Mikhailov, Ekaterina Taktasheva, Elina Sigdel, and Ekaterina Artemova. 2021. RuSentEval: Linguistic source, encoder force! In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pages 43–65, Kiyv, Ukraine. Association for Computational Linguistics.

Aaron Mueller, Garrett Nicolai, Panayiota Petrou-Zeniou, Natalia Talmina, and Tal Linzen. 2020. Cross-linguistic syntactic evaluation of word prediction models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shauli Ravfogel, Yoav Goldberg, and Francis Tyers. 2018. Can LSTM learn to capture agreement? the case of basque. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 98–107, Brussels, Belgium. Association for Computational Linguistics.

Paul Rodriguez and Janet Wiles. 1997. Recurrent neural networks can learn to implement symbol-sensitive counting. In *Advances in Neural Information Processing Systems*, volume 10. MIT Press.

Stuart M. Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343.

Natalia Skachkova, Thomas Trost, and Dietrich Klakow. 2018. Closing brackets with recurrent neural networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 232–239, Brussels, Belgium. Association for Computational Linguistics.

Taiga Someya and Yohei Oseki. 2023. JBLiMP: Japanese benchmark of linguistic minimal pairs. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1581–1594, Dubrovnik, Croatia. Association for Computational Linguistics.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. 2019. LSTM networks can perform dynamic counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence. Association for Computational Linguistics.

Daniela Trotta, Raffaele Guarasci, Elisa Leonardelli, and Sara Tonelli. 2021. Monolingual and cross-lingual acceptability judgments with the Italian CoLA corpus. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2929–2940, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł Ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. Superglue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019c. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, Melbourne, Australia. Association for Computational Linguistics.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN language models learn about Filler–Gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221, Brussels, Belgium. Association for Computational Linguistics.

Ethan Wilcox, Roger P. Levy, and Richard Futrell. 2019. Hierarchical representation in neural language models: Suppression and recovery of expectations. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 181–190, Florence, Italy. Association for Computational Linguistics.

Janet Wiles and Jeffrey Elman. 1995. Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks.

Beilei Xiang, Changbing Yang, Yu Li, Alex Warstadt, and Katharina Kann. 2021. CLiMP: A benchmark for Chinese language model evaluation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2784–2790, Online. Association for Computational Linguistics.