Targeted Syntactic Evaluations with Pseudo-languages

Anonymous ACL submission

Abstract

In this paper, we propose a novel method called **Targeted Syntactic Evaluations with pseudo-languages**, which can (i) test whether language models capture linguistic phenomena without relying on other superficial cues (e.g., lexical information), (ii) control the factors out of interest (e.g., vocabulary size), and (iii) easily test linguistic phenomena that only exist in few languages without collecting corpora of those languages. Specifically, we create four types of pseudo-languages with the abstracted vocabulary of different sizes to control the effect of lexical information and vocabulary size, and with different levels of syntactic complexity: $(\text{Adj})^n$ NP, NP^n VP^n, Nested Dependency, and Cross Serial Dependency. We evaluate four different language models (LSTM, BiLSTM, Transformer Encoder, and Transformer Decoder) on these pseudo-languages, using a binary classification of strings based on their grammaticality. Our result demonstrated that the language models have successfully captured the $(\text{Adj})^n$ NP type phenomenon irrespective of vocabulary size, while they failed to capture the other phenomena as the vocabulary size increases. These results are not consistent with the previous findings that LSTM or Transformer-based language models can capture syntactic dependencies in natural languages to some extent (Hu et al., 2020; Wilcox et al., 2019; Warstadt et al., 2020), suggesting that these language models may not necessarily capture the rules behind these phenomena but rather use some other superficial cues such as co-occurrence or frequency.

1 Introduction

In the field of natural language processing (NLP), language models based on an artificial neural network have achieved remarkable success in various downstream tasks (Wang et al., 2019c,b). 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 artificial neural network-based language models can and cannot capture. The Targeted Syntactic Evaluations first focused on subject-verb number 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).

Although this line of work has provided a useful framework to test the syntactic ability of language models, there are several methodological limitations, mainly because it utilized the sentences sampled from natural corpora or generated from templates (cf., Newman et al., 2021). First, the use of sentences in natural corpora leaves uncertainty as to whether language models solve the problems by truly capturing the rules behind the linguistic phenomena or fraudulently finding other superficial cues (e.g., lexical information such as co-occurrence or word frequency). Second, it also makes it difficult to make comparison across languages; we cannot eliminate factors out of interest (e.g., vocabulary size) when comparing the performances of language models on linguistic phenomena which exist in multiple languages. Third, we cannot easily test linguistic phenomena that only exist in few languages without collecting corpora of those languages and training the language models on them.

To overcome these limitations, we propose a novel method called **Targeted Syntactic Evaluations with pseudo-languages**, which can (i) test whether language models capture linguistic phe-

nomena without relying on other superficial cues (e.g., lexical information), (ii) control the factors out of interest (e.g., vocabulary size), and (iii) easily test linguistic phenomena that only exist in few languages without collecting corpora of those languages. Specifically, we create pseudo-languages with the abstracted vocabulary of different sizes to control the effect of lexical information and vocabulary size, and evaluate language models on a binary classification task of strings based on their grammaticality. In addition, to make our pseudo-languages look more like natural languages, the distribution of the words in our pseudo-languages follows Zipf distribution, which it is claimed the distribution of the words (Zipf, 1942) or the phrases (Ryland Williams et al., 2015) in natural languages follow.

Another methodological novelty of this paper lies in the classification of linguistic phenomena based on their syntactic complexity. In fact, linguistic phenomena can be classified into several classes based on their syntactic complexity (i.e., the Chomsky hierarchy; Chomsky, 1956), but the previous work did not attempt to classify each linguistic phenomenon and exhaustively investigate all of these classes.[1] In this paper, we test four types of pseudo-languages with different levels of complexity : (1) $(\mathrm{Adj})^n$ NP type, which imitates the repetition of adjectives before nouns, (2) NP^n VP^n type, which imitates the embedded sentences without grammatical agreement between noun phrases and verb phrases, as seen in Japanese, (3) Nested Dependency type, which imitates the embedded sentences with the grammatical agreement between noun phrases and verb phrases as seen in English, and (4) Cross Serial Dependency type, which imitates the sentences with multiple dependencies crossing each other, as seen in Swiss German.

For each phenomenon, we create six variants for each pseudo-language with varying vocabulary sizes and test four different language models (LSTM, Hochreiter and Schmidhuber, 1997; BiLSTM, Schuster and Paliwal, 1997; Transformer Encoder; Transformer Decoder, Vaswani et al., 2017) against these pseudo-languages in order to investigate their ability to correctly classify strings based on their grammaticality.

Our result demonstrated that the language models have successfully captured the $(\mathrm{Adj})^n$ NP type phenomenon irrespective of vocabulary size, while they failed to capture the other phenomena as the vocabulary size increases. These results are not consistent with the previous findings that LSTM or Transformer-based language models can capture syntactic dependencies in natural languages to some extent (Wilcox et al., 2019; Warstadt et al., 2020; Hu et al., 2020), suggesting that these language models may not necessarily capture the rules behind these phenomena but rather use some other superficial cues such as co-occurrence or frequency.[2]

2 Experiments

2.1 Pseudo-languages

In this subsection, we introduce the four types of pseudo-languages investigated in this paper: $(\mathrm{Adj})^n$ NP, $\mathrm{NP}^n\mathrm{VP}^n$, Nested Dependency, and Cross Serial Dependency.[3]

Formal definition of pseudo-languages We define the pseudo-languages investigated in this paper as follows: Let $V_{\mathrm{non.}}$ be a set of finite non-terminal symbols. For each non-terminal symbol $A \in V_{\mathrm{non.}}$, 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 $L_{\mathrm{non.}} \subseteq V^*_{\mathrm{non.}}$. Then, each string contained in $L_{\mathrm{non.}}$ 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 $V_{\mathrm{non.}}$, the number T of terminal symbols corresponding to each non-terminal symbol, and the set $L_{\mathrm{non.}}$ of non-terminal symbol strings. Note that $L_{\mathrm{non.}}$ and L belong to the same class in the Chomsky hierarchy: If there is a grammar that generates $L_{\mathrm{non.}}$, we can construct a grammar that generates language L by applying

[1] This classification can be also applied to formal languages, and the literature on formal languages and artificial neural network-based language models (Delétang et al., 2022) conduct unified experiments on all the classes of complexity. Note that the pseudo-languages in this paper are different from the formal languages utilized in this literature, in that ours reproduce the characteristics of natural languages such as the various vocabulary which follows the Zipf distribution, but the formal languages in (Delétang et al., 2022) did not assume finer terminal symbols.

[2] We will make our code publicly available on the acceptance of this paper.

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

the rewriting rules $A \rightarrow a_{A,0}|\cdots|a_{A,T}$ 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 $a_{A,0} \rightarrow A, \cdots, a_{A,T} \rightarrow A$.

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

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

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

This type of pseudo-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 (old)^n man comes.

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

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

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

This pseudo-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 pseudo-language $L_{\text{non.}}$ which belongs to this type is defined as follows:

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

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

This type of pseudo-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 dependency between each NP and VP.

(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:

$$NP \rightarrow \text{Taroo}|\text{Hanako}|\text{Ziroo}|\cdots \quad (7)$$

$$VP \rightarrow \text{hasitta}|\text{itta}|\text{omotta}|\cdots \quad (8)$$

This pseudo-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 pseudo-language $L_{\text{non.}}$ which belongs to this type is defined as follows:

$$V_{\text{non.}} = \{NP_0, \cdots, NP_{N-1}, VP_0, \cdots, VP_{N-1}\} \quad (9)$$

$$L_{\text{non.}} = \{NP_{i_0} \cdots NP_{i_{n-1}} VP_{i_{n-1}} \cdots VP_{i_0} : n \geq 0;\ 0 \leq\ i_0, \cdots, i_{n-1} \leq N-1\} \quad (10)$$

This type of pseudo-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 pseudo-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 pseudo-language $L_{\text{non.}}$ which belongs to this type is defined as follows:

$$V_{\text{non.}} = \{NP_0, \cdots, NP_{N-1}, VP_0, \cdots, VP_{N-1}\} \quad (16)$$

$$L_{\text{non.}} = \{NP_{i_0} \cdots NP_{i_{n-1}} VP_{i_0} \cdots VP_{i_{n-1}} : n \geq 0;\ 0 \leq\ i_0, \cdots, i_{n-1} \leq N-1\} \quad (17)$$

This type of pseudo-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 pseudo-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.

2.2 Data Generation

In this paper, we perform a CoLA-style Targeted Syntactic Evaluation (Warstadt et al., 2019); we evaluate a language model on a binary classification task to predict whether a given string belongs to each of the pseudo-languages defined in the previous subsection. We prepare data for each form language defined in the previous subsection with six different numbers of terminal symbols (= T in section 2.1): 2, 5, 10, 100, 1000, and 5000. This results in a total of 24 pseudo-languages. Note importantly that terminal symbols for a corresponding non-terminal are sampled from Zipf distribution, which it is claimed the distribution of the words (Zipf, 1942) or the phrases (Ryland Williams et al., 2015) in natural languages follow. For simplicity, we only test the Nested Dependency and Cross Serial Dependency types with five non-terminal symbols ($V_{\text{non.}} = \{NP_0, \cdots, NP_4, VP_0, \cdots, VP_4\}$).

2.2.1 Data size and data split

For each pseudo-language, we create 47,500 random samples of strings of length $l \sim \mathcal{U}(4, 30)$. We use 40,000 samples as train data, and 5,000 and 2,500 samples for validation and (in-dist) test data, respectively. We also create "out-of-dist" test data by sampling 2,500 strings of length $l \sim \mathcal{U}(31, 100)$ from each grammar, to evaluate whether the language model can generalize to longer strings than those seen during training. Finally, we generate negative examples for each of the 50,000 positive examples in the manner described in Subsection 2.2.3, and add them to the dataset, resulting in a total of 100,000 examples for each pseudo-language.

2.2.2 Generating positive examples

We generate positive examples for each pseudo-language by first sampling the necessary number of non-terminal symbol sequences belonging to the pseudo-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 Zipf distribution.

Algorithm 1

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 Zipf Distribution
    $string.append(v)$
  end for
  return $string$
end function

2.2.3 Generating negative examples

We generate the negative examples for each data pseudo-language in the following way (cf. Table 1):

(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 positive example of length l.

NP^{n}VP^n We generate a negative example by changing the number of *VP*'s (=n) in a positive example with $m \neq n$ such that the length of the generated negative example will be in the specified length range for each data split.

Data type	Positive example	Negative examples
$Adj^n\ NP$	*Adj Adj Adj Adj Adj NP*	*Adj Adj Adj* **NP** *Adj NP* *Adj* **NP NP NP** *Adj NP*
$NP^n\ VP^n$	*NP NP NP VP VP VP*	*NP NP NP VP VP VP* **VP VP** *NP NP NP* **VP VP**
Nested Dependency	$NP_1\ NP_3\ NP_4\ VP_4\ VP_3\ VP_1$	$NP_1\ NP_3\ NP_4$ **VP**$_2$ $VP_3\ VP_1$ $NP_1\ NP_3\ NP_4$ **VP**$_2$ VP_3 **VP**$_3$
Cross Serial Dependency	$NP_3\ NP_2\ NP_1\ VP_3\ VP_2\ VP_1$	$NP_3\ NP_2\ NP_1$ **VP**$_1$ $VP_2\ VP_1$ $NP_3\ NP_2\ NP_1$ **VP**$_1$ **VP**$_3$ VP_1

Table 1: Examples for each pseudo-language. Here, we use non-terminal symbols and ignore terminal symbols.

Nested/Cross Serial Dependency We generate a negative example of length l by replacing $k \sim \mathcal{U}(1, l/2)$ occurrences of VP_n $(0 \leq n \leq 4)$ in a positive example of length l with VP_m $(0 \leq m \leq 4, m \neq n)$. For these pseudo-languages, 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.

2.3 Models

In this paper, we evaluate the performance of the following four neural language models using the proposed pseudo-languages.

LSTM A single-layer LSTM (Hochreiter and Schmidhuber, 1997) language model with 256-dimensional word embedding and 256-dimensional hidden layer, implemented in PyTorch.[4] The total number of parameters is around 1.05M. We use the hidden state at the last time step for the last layer as the input into the classification layer, which then uses this embedding to produce the grammatical/ungrammatical prediction.

BiLSTM A single-layer BiLSTM (Schuster and Paliwal, 1997) language model with 128-dimensional word embedding and 128-dimensional hidden layer, implemented in PyTorch. The total number of parameters is around 791k. We concatenate the hidden states from both directions at the last time step for the last layer, and use the concatenated vector as the input into the classification layer, which then uses this embedding to produce the grammatical/ungrammatical prediction.

Transformer Encoder A 3-layer, 4-head Transformer (Vaswani et al., 2017) encoder with 128-dimensional word embedding, 1024-dimensional feedforward layer, and sinusoidal word encoding, implemented in PyTorch. The total number of parameters is around 1.32M. We use the average of the embeddings for each input token as the input into the classification layer, which then uses this embedding to produce the grammatical/ungrammatical prediction.

Transformer Decoder A Transformer (Vaswani et al., 2017) decoder with the same configuration as the Transformer Encoder explained above. The only difference is the causal masking applied to the input. We use the embedding for the last input token as the input into the classification layer, which then uses this embedding to produce the grammatical/ungrammatical prediction.

Training and Evaluation Configurations We use SGD for LSTM/BiLSTM models and AdamW (Loshchilov and Hutter, 2019) for Transformer Encoder/Decoder models. 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/BiLSTM models, 0.0001/0.0003/0.0005 for Transformer Encoder/Decoder models).[5] The score for each architecture is defined as the maximum accuracy among the 30 models (10 random seeds $\times$ 3 learning rates) on test data. This is because we are interested in whether these language models can recognize our pseudo-languages at all, and this evaluation metric follows that of Delétang et al. (2022).

[4] https://pytorch.org/

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

Data Type	#Terminals	Transformer Encoder		Transformer Decoder		BiLSTM		LSTM	
		in-dist	out-of-dist	in-dist	out-of-dist	in-dist	out-of-dist	in-dist	out-of-dist
$(\text{Adj})^n$ NP	2	**100.0**	**99.48**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
	5	**100.0**	**99.36**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
	10	**100.0**	**99.66**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
	100	**100.0**	**99.66**	**99.98**	**100.0**	**100.0**	**100.0**	**99.94**	**99.86**
	1000	**99.62**	**99.06**	**98.96**	**99.35**	**99.21**	**99.53**	**99.46**	**99.44**
	5000	**97.29**	**97.76**	**96.73**	**98.34**	**97.36**	**99.01**	**97.76**	**98.76**
NP^nVP^n	2	**100.0**	**96.48**	**100.0**	**100.0**	**100.0**	**100.0**	**99.86**	52.04
	5	**100.0**	**95.90**	**100.0**	**100.0**	**100.0**	**100.0**	**99.84**	51.50
	10	**100.0**	**95.00**	**100.0**	**100.0**	**100.0**	**100.0**	**99.72**	51.44
	100	**99.98**	**97.36**	**98.90**	74.02	**100.0**	**95.16**	**98.40**	73.20
	1000	**99.71**	75.06	77.68	51.98	83.15	53.49	**92.12**	61.92
	5000	**91.33**	50.78	69.62	50.76	76.24	51.44	83.45	56.65
Nested Dependency	2	**99.28**	69.14	**99.20**	**95.88**	**98.72**	**94.68**	**98.82**	66.84
	5	**99.32**	69.66	90.74	80.44	88.82	78.90	**98.20**	65.42
	10	**99.14**	75.74	54.74	55.50	54.56	51.30	**97.48**	68.48
	100	**93.20**	71.89	50.35	50.54	50.68	50.43	74.56	67.63
	1000	51.92	50.22	50.90	50.59	51.28	50.55	50.80	51.08
	5000	50.80	50.49	50.47	50.67	50.66	51.17	50.30	51.08
Cross Serial Dependency	2	**99.38**	73.18	**99.44**	**94.94**	**99.28**	**95.90**	**98.48**	77.34
	5	**99.42**	88.38	**97.00**	90.04	**97.16**	**92.26**	**97.64**	77.54
	10	**99.14**	88.48	83.50	64.68	69.48	58.48	**94.60**	79.48
	100	**91.74**	76.65	50.98	50.43	51.11	50.68	52.44	50.96
	1000	52.19	50.76	50.88	51.25	51.09	50.51	51.18	50.95
	5000	51.18	50.78	51.37	50.69	50.82	51.56	50.85	51.19

Table 2: The accuracy on the in-dist/out-of-dist test data for each pseudo-language and architecture. The accuracy is the maximum achieved by models trained with different 10 random seeds and 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.

3 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. The score for each architecture was obtained by taking the maximum accuracy of 30 models trained with different 10 random seeds and 3 different learning rates. We considered architectures that achieved an accuracy greater than 90% to have successfully captured the rules of each data. In the following sections, we organize the results and provide our analysis for each pseudo-language.

$(\text{Adj})^n$ NP For this type of pseudo-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. Given that our pseudo-language doesn't have any words as seen in natural languages, this result suggests that all the architectures are capable of capturing this syntactic phenomenon without relying on lexical information. Furthermore, the high accuracy observed in this pseudo-language regardless of the number of terminal symbols suggests that each architecture can capture this syntactic phenomenon across languages regardless of the vocabulary size of the language.

NP^n VP^n The Transformer Encoder architecture successfully captured the rules for in-dist test data irrespective of the number of terminal symbols, while other architectures failed as the number of

terminal symbols increased. In the case of out-of-dist test data, some architectures were able to generalize the rules correctly when the vocabulary size is comparatively small, but all architectures failed when the vocabulary size exceeded 1,000. These findings suggest that when lexical information is not available as it is in natural languages, these architectures cannot generalize this syntactic phenomenon when the vocabulary size is large.

Nested Dependency For in-dist test data, Transformer Encoder and LSTM were able to successfully capture the rules when the number of terminal symbols was small, while the other architectures failed in almost all cases. As for out-of-dist test data, some of the architectures were able to generalize the rules correctly in some cases where the number of terminal symbols is small, but almost all of them failed in almost all cases. Notably, the Transformer-based model was not able to capture the rules in most cases regardless of the number of terminal symbols. These results are not consistent with the previous findings that LSTM or 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. Our work is the first to evaluate this phenomenon using a pseudo-language. In terms of the in-dist test data, Transformer Encoder and LSTM were able to successfully capture the rules when the number of terminal symbols was small, while other architectures failed in almost all cases. As for the out-of-dist test data, some of the architectures were able to generalize the rules correctly with a small number of terminal symbols, but no architectures successfully captured the rules when the number of terminal symbols is greater than 5. Notably, the Transformer-based model was unable to capture the rules in all cases regardless of the number of terminal symbols.

Summary The language models only exhibited an ability to capture the structural patterns of the $(\text{Adj})^n$ NP type pseudo-language even with an increased vocabulary size. While previous studies claimed that language models are able to capture certain syntactic structures, such as nested dependency, to some extent (Hu et al., 2020; Wilcox et al., 2019), our results suggest that the language models may have actually relied on superficial cues, such as lexical semantics or co-occurrence frequency of words, to solve the tasks. Furthermore, although this study only focused on a limited number of language phenomena, the results also suggest that the models may not be able to solve other grammatical phenomena when controlling for lexical information. Therefore, the conventional approach of Targeted Syntactic Evaluations using texts from natural corpora may not have achieved its original goal of measuring the pure syntactic ability of language models. To achieve this, new methods such as the one introduced in this study may be necessary.

4 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. However, it is impossible to compare the performance of language models across languages without identifying shared linguistic phenomena across multiple languages. In addition, even if such shared phenomena are identified and a comparison is conducted, it remains challenging to determine the linguistic characteristics responsible for variations in performance and to isolate the impact of performance differences in the language models used. Furthermore, it is difficult to evaluate rare linguistic phenomena that only exist in a specific language. Our proposed methodology, utilizing pseudo languages, provides a potential solution to these challenges.

Another line of research has focused on the ability of neural language models to recognize formal languages (Bodón and Wiles, 2000; Boden and Wiles, 2002; Suzgun et al., 2019; Bhattamishra et al., 2020; Ebrahimi et al., 2020; Hahn, 2020; Delétang et al., 2022). These studies have investigated, both mathematically 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, as correctly pointed out by Delétang et al. (2022), just because a language model recognizes one language on a certain level of the Chomsky hierarchy, it cannot be guaranteed that there are no other languages on the same level that the language model cannot recognize. From this perspective, this study can also be considered as an attempt to examine what differences, if any, arise from assuming finer terminal symbols corresponding to the vocabulary of natural languages compared to the results obtained in previous studies.

5 Conclusion

In this paper, we proposed a novel method called **Targeted Syntactic Evaluations with pseudo-languages**, which can (i) test whether language models capture linguistic phenomena without relying on other superficial cues (e.g., lexical information), (ii) control the factors out of interest (e.g., vocabulary size), and (iii) easily test linguistic phenomena that only exist in few languages without collecting corpora of those languages. Specifically, we created pseudo-languages with abstracted vocabulary of different sizes to control the effect of lexical information and vocabulary size, and evaluated language models on a binary classification task of strings based on their grammaticality. We tested four types of pseudo-languages with different levels of syntactic complexity: $(\text{Adj})^n$ NP, NP^nVP^n, Nested Dependency, and Cross Serial Dependency. Our result demonstrated that the LSTM and Transformer-based models can successfully capture the $(\text{Adj})^n$ NP type phenomenon irrespective of vocabulary size, while they failed to capture the other phenomena as the vocabulary size increases. These results are not consistent with the previous findings that LSTM or Transformer-based language models can capture syntactic dependencies in natural languages to some extent (Hu et al., 2020; Wilcox et al., 2019; Warstadt et al., 2020), suggesting that these language models may not necessarily capture the rules behind these phenomena but rather use some other superficial cues such as co-occurrence or frequency. We will leave it for future research to expand the variety of pseudo-languages and language models.

Limitations

There are several limitations with this paper. First, the models tested were limited in their variety. To evaluate whether each architecture was able to capture the rules of each pseudo-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. Second, the method for creating negative examples for each pseudo-language is limited. While we used a specific method to generate negative examples for each pseudo-language, there are countless other ways to generate negative examples. It is unclear whether the results of this study hold true with other types of negative examples.

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.

N. Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124.

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. 2022. Neural networks and the chomsky hierarchy.

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*.

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.

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.

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.

Benjamin Newman, Kai-Siang Ang, Julia Gong, and John Hewitt. 2021. Refining targeted syntactic evaluation of language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3710–3723, Online. 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.

Jake Ryland Williams, Paul R. Lessard, Suma Desu, Eric M. Clark, James P. Bagrow, Christopher M. Danforth, and Peter Sheridan Dodds. 2015. Zipf's law holds for phrases, not words. *Scientific Reports*, 5(1):12209. Number: 1 Publisher: Nature Publishing Group.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

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.

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.

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.

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.

George Kingsley Zipf. 1942. The unity of nature, least-action, and natural social science. *Sociometry*, 5(1):48–62.

A Additional Results

In section 3, we reported the results when the distribution of terminal symbols follows Zipf distribution. Here, we report the results when the distribution is uniform. As shown in Table 3, similar results were obtained when using the uniform distribution compared to when using the Zipf distribution. This suggests that the language model is able to capture certain syntactic phenomena regardless of the distribution of the vocabulary.

Pseudo-language	#Terminals	Transformer Encoder		Transformer Decoder		BiLSTM		LSTM	
		in-dist	out-of-dist	in-dist	out-of-dist	in-dist	out-of-dist	in-dist	out-of-dist
$(\text{Adj})^n$ NP	2	**100.0**	**99.48**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
	5	**100.0**	**99.36**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
	10	**100.0**	**99.66**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
	100	**100.0**	**99.66**	**99.98**	**100.0**	**100.0**	**100.0**	**99.94**	**99.86**
	1000	**99.62**	**99.06**	**98.96**	**99.35**	**99.21**	**99.53**	**99.46**	**99.44**
	5000	**97.29**	**97.76**	**96.73**	**98.34**	**97.36**	**99.01**	**97.76**	**98.76**
NP^nVP^n	2	**100.0**	**96.48**	**100.0**	**100.0**	**100.0**	**100.0**	**99.86**	52.04
	5	**100.0**	**95.90**	**100.0**	**100.0**	**100.0**	**100.0**	**99.84**	51.50
	10	**100.0**	**95.00**	**100.0**	**100.0**	**100.0**	**100.0**	**99.72**	51.44
	100	**99.98**	**97.36**	**98.90**	74.02	**100.0**	**95.16**	**98.40**	73.20
	1000	**99.71**	75.06	77.68	51.98	83.15	53.49	**92.12**	61.92
	5000	**91.33**	50.78	69.62	50.76	76.24	51.44	83.45	56.65
Nested Dependency	2	**99.28**	69.14	**99.20**	**95.88**	**98.72**	**94.68**	**98.82**	66.84
	5	**99.32**	69.66	**90.74**	80.44	88.82	78.90	**98.20**	65.42
	10	**99.14**	75.74	54.74	55.50	54.56	51.30	**97.48**	68.48
	100	**93.20**	71.89	50.35	50.54	50.68	50.43	74.56	67.63
	1000	51.92	50.22	50.90	50.59	51.28	50.55	50.80	51.08
	5000	50.80	50.49	50.47	50.67	50.66	51.17	50.30	51.08
Cross Serial Dependency	2	**99.38**	73.18	**99.44**	**94.94**	**99.28**	**95.90**	**98.48**	77.34
	5	**99.42**	88.38	**97.00**	**90.04**	**97.16**	**92.26**	**97.64**	77.54
	10	**99.14**	88.48	83.50	64.68	69.48	58.48	**94.60**	79.48
	100	**91.74**	76.65	50.98	50.43	51.11	50.68	52.44	50.96
	1000	52.19	50.76	50.88	51.25	51.09	50.51	51.18	50.95
	5000	51.18	50.78	51.37	50.69	50.82	51.56	50.85	51.19

Table 3: The accuracy on the test data for each data type and architecture. The accuracy is the maximum achieved by models trained with different 10 random seeds and 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. The distribution of the terminal symbols is uniform.