What do Large Language Models Learn beyond Language?

Avinash Madasu **Shashank Srivastava**
UNC Chapel Hill
{avinashm,ssrivastava}@cs.unc.edu

Abstract

Large language models (LMs) have rapidly become a mainstay in Natural Language Processing. These models are known to acquire rich linguistic knowledge from training on large amounts of text. In this paper, we investigate if pre-training on text also confers these models with helpful 'inductive biases' for non-linguistic reasoning. On a set of 19 diverse non-linguistic tasks involving quantitative computations, recognizing regular expressions and reasoning over strings. We find that pretrained models significantly outperform comparable non-pretrained neural models. This remains true also in experiments with training non-pretrained models with fewer parameters to account for model regularization effects. We further explore the effect of text domain on LMs by pretraining models from text from different domains and provenances. Our experiments surprisingly reveal that the positive effects of pre-training persist even when pretraining on multi-lingual text or computer code, and even for text generated from synthetic languages. Our findings suggest a hithertho unexplored deep connection between pre-training and inductive learning abilities of language models[1].

1 Introduction

Pretrained Language Models (LMs) have shown singular succcess on a range of natural language understandings tasks, to the extent that they have become foundational for contemporary NLP systems. Several works have investigated why pretraining works so well (Warstadt et al., 2019; Zhao et al., 2020). In particular, studies have shown that the pretrained LMs like BERT capture linguistic knowledge about syntax (Lin et al., 2019; Wu et al., 2020), semantics (Vulić et al., 2020b,a) and morphology (Hofmann et al., 2020, 2021). In fact, Tenney et al. (2019) demonstrated that learned representations

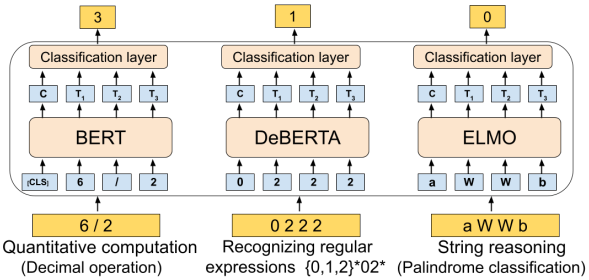


Figure 1: We investigate the effect of pretraining of languages models on learning non-linguistic tasks using three task paradigms involving symbolic reasoning.

in pretrained LMs even internally reflect the classical NLP pipeline. Since most NLP benchmarks such as SuperGLUE (Wang et al., 2019) naturally are focused on tasks such as textual entailment and reading comprehension that require linguistic knowledge and reasoning, it is unsurprising that LMs have achieved strong results on these tasks. On the other hand, little work so far has explored the abilities of pretrained LMs for learning non-linguistic tasks.

In this paper, we explore whether pretraining on text is inherently about learning language, or if pretraining also imbues LMs with skills for symbolic manipulation and non-linguistic reasoning (for example, performing quantitative computation such as finding the median of a set of numbers, recognizing regular expressions, or identifying whether a string is a palindrome, as shown in Figure 1). In other words, we investigate whether and how pretraining develops helpful inductive biases for non-linguistic reasoning. For this analysis, we create a set of 19 tasks from three categories of task paradigms: quantitative computation (§3.1), recognizing regular expressions (§3.2), and string reasoning (§3.3). Figure 1 shows an example for each category, and the full list of tasks is described in the table 1. We experiment with transformer and RNN based LMs (§4) for learning these tasks, and per-

[1] https://github.com/avinashsai/NILM

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 6969 - 6982
December 7-11, 2022

Task	Input Eg.	Output Eg.	Classes	Input range
Odd classification	*4210*	0	0 - 1	[1, 20000]
Even classification	*4210*	1	0 - 1	[1, 20000]
Odd even classification	*4210 even*	1	0 - 1	[1, 20000]
Decimal operation	*872 / 436*	2	0 - 9	[1, 10000]
Decimal & word operation	*four / 2*	2	0 - 9	[1, 10000]
Mean	*15,-8,15,-5,-14,-3 ?*	0	0 - 9	[-15, 15]
Median	*3,6,5,15,2,3,-6,-2,9,-3,-9,-5,-14 ?*	2	0 - 9	[-15, 15]
Mode	*5,9,7,0,2,5,3,3,3,0 ?*	3	0 - 9	[0, 9]
Recognize {0, 1, 2}*02*	*0 1 2 0 2 1 0 2 2 2 2*	1	0 - 1	[0, 2]
Recognize AA*BB*CC*DD*EE*	*a a a a a a a b b b b c c c c c d d d e*	1	0 - 1	[a, e]
Palindrome classification	*a W X X W a*	1	0 - 1	[a-z], [A-Z]
Anagram classification	*r G r P J h k - k h G r P J r*	1	0 - 1	[a-z],[A-Z]
Isogram classification	*v F J o S j*	1	0 - 1	[a-z], [A-Z]
Tautonym classification	*s t P v g - t P v g a*	1	0 - 1	[a-z], [A-Z]
Length of a string	*t e e o*	4	0 - 9	[a-z]
Count of unique characters	*d e i i e e d i i d*	3	0 - 9	[a-j]
Parity check	*0 1 1 1 0 1 0 0 1 1 1 0*	0	0 - 1	[0, 1]
Vowels classification	*i i v x c m o o u o*	0	0 - 9	[a-z]
Maximum frequent character	*j j j c j j*	9 (j)	0 - 9	[a-j]

Table 1: Description of the non-linguistic tasks with input and output examples. Classes are the class labels for each task. Input range denotes the range of the input operands in each task.

form a comparative analysis with (non-pretrained) neural model variants from the perspective of learning metrics such as accuracy and sample efficiency.

Our experiments (§5) reveal that pretrained models overall perform substantially better and are more sample efficient on most tasks. However, there are significant differences and patterns in performance between task types, as well as variance between different LM architectures. Since non-pretrained models do not have the benefit of regularization that comes from pretraining, a plausible reason for the discrepancy between them and pretrained LMs might be underfitting of the non-pretrained models when trained on comparatively small dataset sizes. To account for this, we also comprehensively explore the effect of model size (§6) of non-pretrained models for both transformer and RNN architectures. We find that the discrepancy in performance remains even for smaller neural models, indicating that the differences are not simply due to a mismatch in model and data sizes.

Finally, we investigate the role that pretraining data plays in influencing task performance on non-linguistic tasks (§7). We experiment with pretraining on different domains of text, pretraining on perturbed representations of natural language text (such as shuffled word order), pretraining on text of computer programs (no linguistic properties of natural languages), pretraining on multi-lingual and non-English text, and pretraining with synthetic text (data sampled from synthetic distributions). Our analysis reveals that the advantages of pretraining surprisingly persist with various degrees across these variations, suggesting hithertho unexplored connections between pretraining and the learning abilities of language models. Our contributions are:

- We compare a range of pretrained LMs and non-pretrained models on a carefully designed suite of 19 classifications tasks that require non-linguistic reasoning.
- We comprehensively explore the role of the pretraining data by experimenting with models pretrained from texts with different provenances.
- We establish that the positive effects of pretraining are not simply due to better model regularization by experimenting with neural models with different complexities and architectures.

2 Related Work

A body of work has investigated contextual word embeddings to determine whether they capture aspects of mathematical meaning for numbers (Naik et al., 2019). Wallace et al. (2019) probed numerical supremacy on token embeddings of contextual language models such as ELMO and BERT. (Thawani et al., 2021) surveyed numerical understanding in NLP models using 7 sub-tasks such as measurement estimation and word problems. Our work diverges from these in exploring a richer set of tasks including harder tasks such as set operations. Further, previous methods explore mathematical reasoning tasks posed as language problems, which

conflates the problems of language and mathematical learning and also makes the datasets susceptible to biases due to data collection. Our analysis circumvents both these issues by design.

Some previous works have explored the ability of RNN and Transformer architectures for learning regular languages (Weiss et al., 2018; Sennhauser and Berwick, 2018; Suzgun et al., 2019b; Bhattamishra et al., 2020), closing brackets (Skachkova et al., 2018), and dynamic counting (Suzgun et al., 2019a). However, they focus on the learnability of these tasks with specific architectures, and do not look at pretrained LMs, which are our focus here.

Finally, in our discussion, we conceptually stretch the notion of inductive bias. The idea of inductive bias is usually associated with specific model types (McCoy et al., 2020; Kharitonov and Chaabouni, 2021), architectures (Xu et al., 2021; Brutzkus and Globerson, 2021) and regularization approaches (Helmbold and Long, 2015). We believe that extending this to refer to learning tasks with pretrained LMs is both reasonable and useful.

3 NILM

In this section, we describe the tasks used for our analysis, which we refer to as `NILM` (measuring Non-linguistic Inductive bias in Language Models). The tasks correspond to three task paradigms: (1) quantitative computation, (2) regular expressions, and (3) string reasoning. Each task in `NILM` is posed as a classification task. The descriptions for all the tasks with input and output examples, class labels and the input range are shown in Table 1. Each task has a synthetically generated dataset with train/dev/test splits[2]. To avoid biases in the datasets, relevant numbers and strings in individual examples are uniformly sampled from the appropriate ranges.

3.1 Quantitative computation

This task paradigm focuses on tasks involving arithmetic and set statistics.

Odd classification. Classify if a number is odd.

Even classification. Classify if a number is even.

Odd even classification. For a given number N and a string "even" or "odd", classify if the number satisfies the string condition.

Decimal operation. Subtract or divide two numbers. Operands are represented in decimal notation.

Decimal & word operation. Subtract or divide two numbers. Operands are represented in decimal or word notation.

Mean. Given a set of numbers, output the mean.

Median. Given a set, output the median.

Mode. Given a set of numbers, output the mode.

3.2 Recognizing regular expressions

This task paradigm focuses on recognizing regular expressions. The training data consists of positive and negative examples of strings matching a regular expression (Bhattamishra et al., 2020).

Recognize {0,1,2}*02*. Recognize if a pattern matches {0,1,2}*02*. The maximum length of the patterns is 20.

Recognize AA*BB*CC*DD*EE*. Recognize if a pattern matches AA*BB*CC*DD*EE*. The maximum length of the patterns is 30.

3.3 String reasoning

This task paradigm focuses on reasoning tasks over individual strings or pairs of strings.

Palindrome classification. A string is a palindrome if it reads the same forward and backward. The task is to classify whether a given string is a palindrome. The string length ranges from 1 to 15.

Anagram classification. Two strings are anagrams if one is formed by rearranging letters from the other. The task is to classify if a pair of strings are anagrams. The string length ranges from 2 to 15.

Isogram classification. A string is an isogram if it has no repeating characters. The task is to classify whether a given string is an isogram. The string length ranges from 1 to 52.

Tautonym classification. A tautonym is a word which can be broken down into two identical parts, with the same spelling. The task is to classify whether a given string is a tautonym. The string length ranges from 1 to 10.

Length of a string. Output the length of a given string. The string length ranges from 1 to 10.

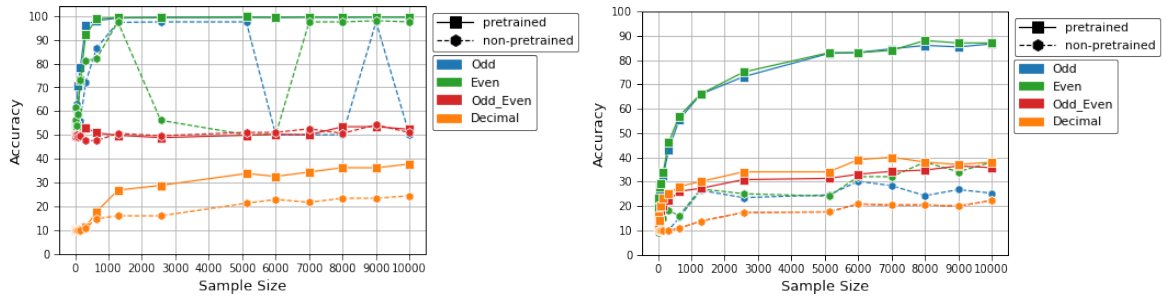
Count of unique characters. Given a string, count the number of unique characters in it. The string lengths ranges from 10 to 30.

Parity check. Given a binary string, output if the counts of ones and zeros are the same. The maximum length of the binary string is 20.

Vowels classification. Given a string, classify if the string contains only vowel characters. The string length ranges from 3 to 10.

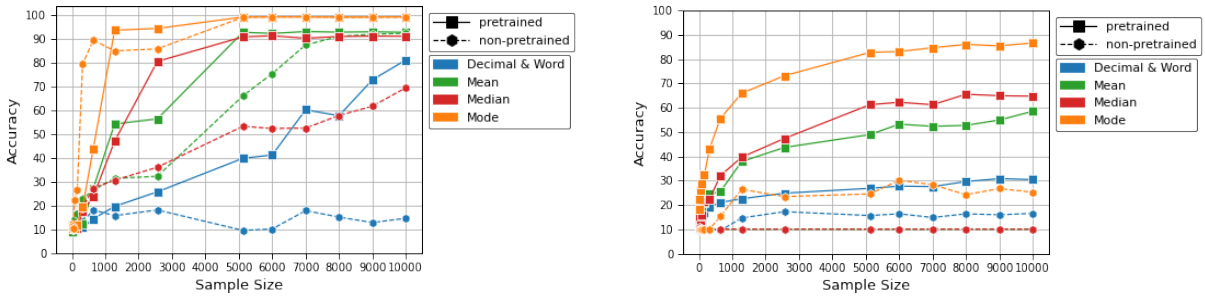
Maximum frequent character. Given a string, output the character with the maximum frequency.

[2]The training set size for all tasks is 10K, dev set size is 1K and test set size is 1K, except for tasks on recognizing regular expressions, where the test set size is 2K following previous work (Bhattamishra et al., 2020).



(a) BERT small (b) ELMO

Figure 2: Performance comparison of pretrained and non-pretrained models of BERT small, and ELMO on four quantitative computation tasks (odd classification, even classification, odd even classification and decimal operation).



(a) BERT small (b) ELMO

Figure 3: Performance comparison of pretrained and non-pretrained models of BERT small, and ELMO on four quantitative computation tasks (mean, median, mode and decimal & word operation tasks).

The string length ranges from 5 to 30.

4 Models & variants

Next, we describe the LMs and their variants used in NILM. We experiment with four language models, based on both Transformer and RNN architectures.
BERT small. This is the bert-base-uncased model with 12 transformer encoder layers and the dimension of the representations is 768. BERT tokenizer is based on the WordPiece model (Wu et al., 2016).
BERT large. This is the bert-large-uncased model which has 24 transformer encoders and representations have 1024 dimensions.
DeBERTa. This is a transformer based language model and its tokenizer is built using Byte Pair Encoding (Sennrich et al., 2016). We consider the DeBERTa base model. It has 12 transformer encoder layers and representations have 768 dimensions.
ELMO. This is an LSTM based language model (Peters et al., 2018). It has 3 layers and the output representations have 1024 dimensions.

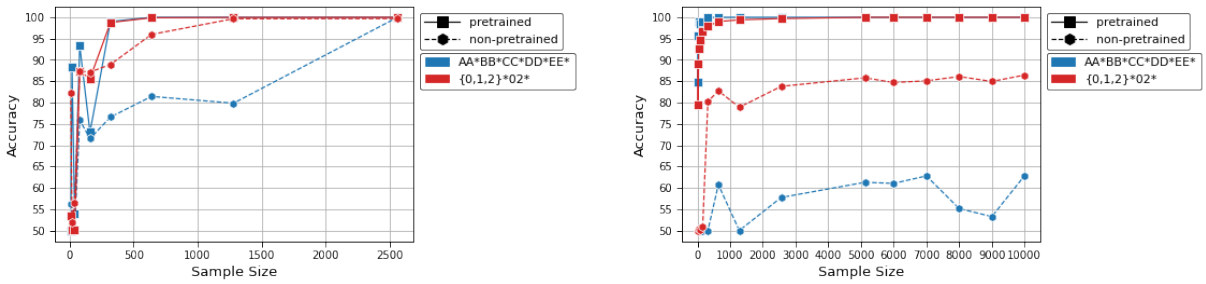
Our experiments are based on pretrained and non-pretrained variants of these architectures. For pretrained variants, the weights are initialized with the pretrained weights. The tokenization on the training data is performed using the pre-built vocabulary. For the non-pretrained neural models, the weights are initialized randomly and updated during training. The tokenizer used is the same as in the pretrained variant.

All the models are trained with varying training data of sizes 10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 6000, 7000, 8000, 9000 and 10000. For training set sizes of less than 1000 samples, we report the average of 10 runs. For training set sizes greater than 1000, all reported numbers are averages of 5 runs. In the next section, we present a comparative analysis of pretrained and non-pretrained models.

5 Comparative Evaluation

Next, we compare the performance of pretrained and non-pretrained models on tasks in NILM[3].
Quantitative computation: Figure 2 shows results on odd classification, even classification, odd even classification and decimal operation tasks. We find that pretrained LMs outperformed non-pretrained model for all of these tasks. Further, Transformer-

[3]Details, including statistical significance results with the paired t-value test, are included in Appendix 6



(a) BERT small

(b) ELMO

Figure 4: Performance comparison of pretrained and non-pretrained models of BERT small, and ELMO on regular expression tasks (AA*BB*CC*DD*EE* and recognize {0,1,2}*02*).

based LMs outperformed the RNN-based ELMO models in all the tasks[4]. We note that for the relatively easy tasks such as odd and even classifications, the pretrained LMs show more stable training. However, for harder tasks such as Decimal operations (where the baseline performance is around 10%), no models are able to learn the task well even with 10K labeled examples.

Figure 3 shows results on median, mean, mode and decimal & word operation tasks. The median task requires complex reasoning (sorting numbers and computing the middle element), and shows significantly lower performance than the mean and mode tasks for the non-pretrained models even with the maximum training set size. The pretrained LM models show little eventual difference in performance between these three tasks. On the other hand, for the easiest of these tasks (mode), non-pretrained models actually show higher performance than pretrained LMs in the low data regime.

Recognizing regular expressions: Figure 4 shows the comparative performance of pretrained LMs on non-pretrained models on the two tasks involving recognizing regular expressions. For both tasks, we note that the pretrained LMs can perfectly learn the tasks with many fewer labeled examples compared to the non-pretrained models. In both cases, the non-pretrained Transformer-based models eventually reach optimal performance as well. However, curiously the ELMO based non-pretrained models struggle with learning both tasks.

String reasoning: Figures 6 show the results on Palindrome, Anagram, Isogram and Tautonym classification. These tasks require character comparison within the string or with another string. Again, the pretrained variants consistently outperformed non-pretrained models variants in all of these tasks. In particular, the non-pretrained models completely fail to learn the Anagram and Palindrome tasks even for the largest training set size. Again, Transformer based LMs outperform LSTM based LMs.

Figure 7 shows the results on vowels classification, maximum frequent character, length of a string and parity check tasks. These tasks don't require intra-string comparisons. We see that most Transformer-based variants eventually achieve optimal performance. For these simpler tasks, we again observe several instances where the Transformer-based non-pretrained models actually outperform pretrained LMs in the low data regime.

6 Effect of model size

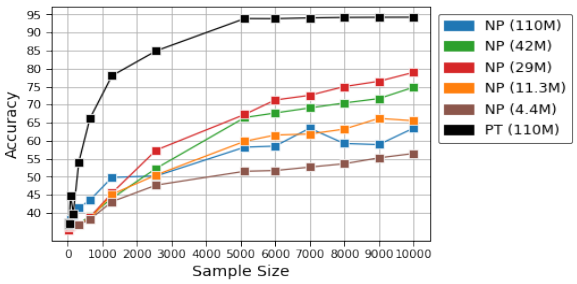


Figure 5: Effect of model size on non-pretrained models. NP denotes a non-pretrained model and PT denotes the pretrained model. Mid-sized non-pretrained models outperform bigger and smaller variants, but still perform significantly lower than pretrained LM models. Results are the average of six representative tasks: palindrome classification, anagram classification, isogram classification, tautonym classification, mean and median.

As previously mentioned, a possible explanation for the underperformance of non-pretrained models ise that the large number of parameters of the

[4]We will focus on BERT small as representative of transformer models. Results for BERT large and DeBERTa follow similar trends, and are included in the supplementary material

architecture relative to the sizes of the training data might be leading to under-fitting. To test this, we experiment with smaller Transformer-based models with varying numbers of parameters.

Figure 5 illustrates the effect of model sizes of non-pretrained model. The original 110 million parameter model has 12 encoder layers, 12 attention heads, and 768 dimensional representations. The 42 million parameter model has 8 encoder layers, 8 attention heads and 512 dimensional representations. The 29 million parameter model has 4 encoder layers, 8 attention heads and 512 dimensional representations. The 11 million parameter model has 4 encoder layers, 4 attention heads and 256 dimensional representations. The smallest 4 million parameter model has 2 encoder layers, 2 attention heads and 128 dimensional representations.

As seen in the figure, reducing the model size significantly improves the average performance of the non-pretrained models over 6 representative tasks. However, the smallest models show a performance drop. Most significantly, even the best performing intermediate-sized architectures are significantly worse than the pretrained LM models. This strongly suggests that the discrepancy between pretrained and non-pretrained models is not simply due to a mismatch between model and data sizes.

7 Effects of Pretraining Data

We observe that pretrained LMs consistently performed better than non-pretrained models. This leads to the natural question of what role the text data used for pretraining plays in the process. Next, we investigate this in depth by experimenting with language models pretrained on different types of text. For this, we pretrain models using the BERT-small and DeBERTa architectures and an MLM objective on different text datasets, and evaluate the performance of these models on `NILM` tasks.

7.1 Variance with text domain

We first explore models pretrained on three different domains of text.

SNLI. We pretrained BERT small from scratch on SNLI data (Bowman et al., 2015). It has 1000k sentences (570k pairs of text and hypothesis).

Amazon reviews. We selected 500k movies and tv reviews from the larger Amazon reviews dataset (He and McAuley, 2016) and used for pretraining. Since reviews are in a free-text format, and their collection was not tailored with a NLP task in mind, they might be more representative of the complexity of real-world language use than SNLI.

ROC. ROC is a corpora of 100K children stories, each made up of five sentences (Mostafazadeh et al., 2017). The language in ROC is relatively simple in both vocabulary and sentence structure.

Tables 2 and 3 shows the average accuracy of six non-linguistic tasks (palindrome classification, isogram classification, tautonym classification, odd even classification, decimal operation and median) fine-tuned using different BERT and DeBERTA representations respectively. We note that the models pretrained on all three domains outperformed the non-pretrained model (NP). This suggests that the results of experiments in Section 5 generalize to new text corpora for pretraining, and do not rely on having access to text on specific topics during pretraining. This is a non-trivial result, since it suggests for example, that the higher performance of pretrained models on tasks such as palindrome and anagram classification is not due to the pretrained models having seen information about such concepts during pretraining. This is especially so since the results even generalize to ROC stories, which contain no information on such technical concepts.

7.2 Perturbed text

Next, we experiment with perturbing the text used for pretraining by changing the order of words in the text. We explore the following models:

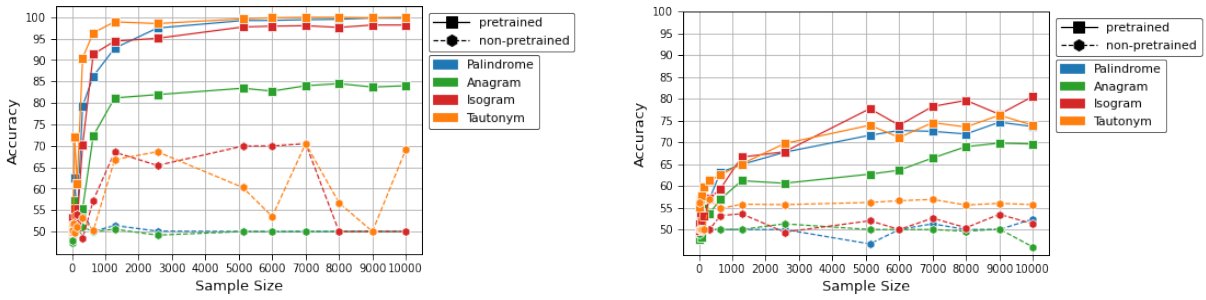
SNLI sort. The words in the sentences of SNLI dataset are sorted based on alphabetical order.

SNLI shuffle. We randomly shuffle words in sentences in the SNLI dataset.

Amazon reviews sort. Similar to SNLI sort, the words in sentences are alphabetically sorted.

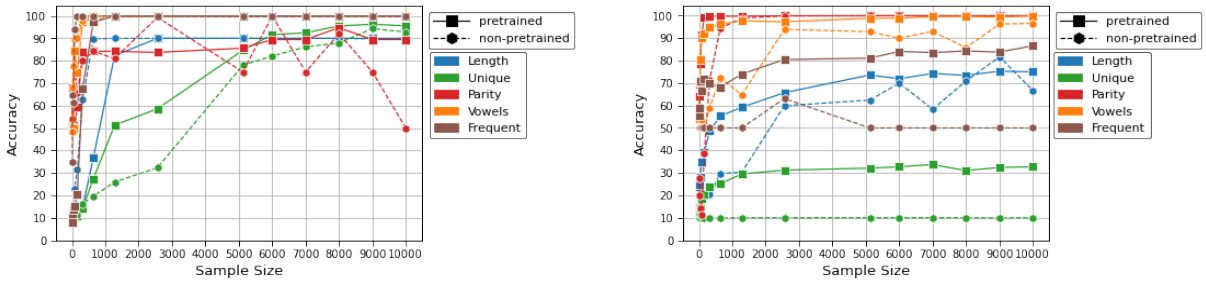
Amazon reviews shuffle. We randomly shuffle words in sentences in the Amazon reviews dataset.

We observe that models pretrained with perturbed text also significantly outperformed non-pretrained models, and perform comparably to the original pretrained LMs. For the SNLI dataset, there is 3% drop in best performance when pretrained on SNLI sort and 2% drop in performance when pretrained on SNLI shuffle for BERT (Table 2). In fact, for DeBERTa, SNLI shuffle outperformed the standard SNLI by 2% (Table 3). Similarly, the Amazon sort and Amazon shuffle versions outperformed or achieved similar performance as the standard Amazon data version. A likely explanation for this is that, even though syntactic word order is disturbed



(a) BERT small (b) ELMO

Figure 6: Performance comparison of pretrained and non-pretrained models of BERT small, and ELMO on four string reasoning tasks (palindrome, anagram, isogram and tautonym classification).



(a) BERT small (b) ELMO

Figure 7: Performance comparison of pretrained and non-pretrained models of BERT small, and ELMO on five string reasoning tasks (length of a string, maximum frequent character, vowels classification, parity check and count of unique character).

Sample size	SNLI	SNLI sort	SNLI shuffle	Amz	Amz sort	Amz shuffle	ROC	X-ling BERT	Chinese BERT	Code BERT	Zipf	Unif	Syn Voc	NP
10	37	39	38	36	36	36	36	38	38	37	38	36	36	37
20	37	37	37	36	38	38	38	37	37	38	37	37	37	37
40	37	38	36	37	36	36	36	42	42	37	42	36	37	37
80	38	40	40	37	38	38	38	55	55	47	55	36	36	38
160	38	40	37	37	40	40	40	56	56	37	56	37	37	39
320	40	49	41	38	41	41	41	64	64	61	64	39	37	41
640	44	60	47	43	52	52	52	75	75	69	75	42	39	44
1280	60	71	63	55	69	69	69	80	80	92	80	52	41	50
2560	76	84	75	75	79	79	79	81	81	89	81	59	48	50
5120	82	87	82	83	89	89	89	94	94	97	94	71	58	58
6000	83	87	83	85	90	90	90	94	94	96	94	73	60	59
7000	88	89	88	89	91	91	91	94	94	97	94	78	62	64
8000	89	89	88	90	92	92	92	94	94	97	94	81	63	59
9000	90	90	89	91	92	92	92	94	94	97	94	84	64	59
10000	91	88	89	91	92	92	92	94	94	97	94	85	64	64

Table 2: Average accuracy scores of different pretrained BERT representations on six representative non-linguistic tasks: palindrome, anagram, isogram, tautonym, mean, and median. The results are rounded to the nearest percentage point. All models except Synthetic Vocabulary (Syn Voc). show statistically significant improvements ($p < 0.05$) over the non-pretrained models.

by shuffling, distributional information over sentence contexts is still preserved in the perturbed data. We describe experiments with text data having no distributional information in later sections.

7.3 Non-English and Computer Languages

A possible rationale for explaining the beneficial effect of pretraining for non-linguistic tasks is that irrespective of whether the tasks require non-linguistic reasoning, their *format* is in language, and hence language models should be able to learn these tasks with fewer examples. To test this hypothesis, we also experiment with models pretrained on text from languages different from English, as well as models pretrained on computer code. These include the following models:

Multilingual BERT. Multilingual BERT is pretrained on text from 102 different languages. About

Sample size	SNLI	SNLI sort	SNLI shuffle	Amz	Amz sort	Amz shuffle	ROC	X-ling DeBERTa	Zipf	Unif	Syn Voc	NP
10	36	36	37	36	35	36	37	36	37	36	36	37
20	37	36	36	36	35	35	37	39	36	37	37	37
40	37	36	36	36	36	35	37	38	37	36	37	37
80	38	37	39	37	37	36	37	38	37	36	36	37
160	37	38	37	36	38	37	37	40	38	37	37	38
320	39	39	39	37	42	39	41	58	40	39	37	38
640	44	44	45	42	52	46	48	71	47	42	39	47
1280	54	51	54	50	72	58	52	80	61	52	41	60
2560	70	70	69	65	81	72	65	90	75	59	48	72
5120	79	78	80	79	87	83	83	93	83	71	58	73
6000	79	82	80	81	88	84	82	91	84	73	60	74
7000	84	86	87	85	89	87	84	93	84	78	62	74
8000	85	87	87	86	89	88	85	93	87	81	63	76
9000	86	87	88	86	91	90	85	93	88	84	64	77
10000	87	87	89	86	91	90	85	93	87	85	64	78

Table 3: Average accuracy scores of different pretrained DeBERTA representations on six representative non-linguistic tasks: palindrome, anagram isogram, tautonym, mean, and median. The results are rounded to the nearest percentage point. All models except Synthetic Vocabulary (Syn Voc). show statistically significant improvements ($p < 0.05$) over the non-pretrained models.

21% of the pretraining text is English.

Chinese BERT. Chinese BERT is a BERT model pretrained on Chinese text.

Code BERT. CodeBERT (Feng et al., 2020) is pretrained on code from six programming languages.

In Table 2, we note that all three non-English pretrained LMs significantly outperformed non-pretrained models, with the best performance being comparable or marginally lower than English versions. In fact, Code-BERT surprisingly surpasses ROC by 5%. These findings strongly indicate that the advantages from pretraining have little to do with the format of the tasks, since they persist for scenarios with little shared linguistic structure.

7.4 Synthetic languages

Finally, to investigate what happens if we weaken the distributional properties that hold even in the perturbed text versions from Section 6.2, we experiment with pretraining models on synthetic text sampled from simple probability distributions:

Zipf distribution. We select 30k words (types) from the Amazon reviews dataset. Words are picked with a unigram probability that follows Zipf's word frequency law, which all natural languages empirically follow (Piantadosi, 2014). For the Zipf distribution, we chose α=1 and β=2.7, to match the parameters of most natural languages. The text does not follow any word order.

Uniform distribution. In this dataset, words are sampled from the same vocabulary as in 'Zipf distribution', but with a uniform unigram probability. The text does not follow any word order.

Synthetic Vocabulary. Words are selected with uniform distribution from a vocabulary to form sentences. However, instead of a vocabulary of English words, the words in the vocabulary are also synthetically generated (3 letter combinations of lower-case alphabets). In this text, the words do not possess morphology in addition to no syntax.

In Tables 2 and 3, we note that surprisingly, even models pretrained on Zipfian and uniform distribution text continue to outperform the non-pretrained models. In fact, the Zipf version's best accuracy is 3% higher than the standard Amazon data version and 2% compared to perturbed Amazon shuffled data version in case of BERT. Zipf outperforms standard amazon data by 1% and lags behind amazon shuffle by 3% for DeBERTA. The Uniform distribution version lags behind Zipf by 9% and 2% for BERT and DeBERTa respectively. We note that the Zipf and Uniform versions still use the prebuilt vocabulary from the Amazon data, and hence this text maintains morphological structure. However, the gains finally disappear for the Synthetic vocabulary model, which cannot leverage morphological structure in the text, and its performance is similar to the non-pretrained models.

8 Conclusion

We explore the non-linguistic inductive biases of pretrained LMs. While the general trend (that pretraining helps) is unsurprising, our analysis with models pretrained on different text corpora shows that this is not due to the model seeing related topics during pretraining. We find that these gains persist even in absence of any shared linguistic structure (in cross-lingual settings). Our observation that this behavior is seen even when pretraining on synthetically generated languages is intriguing

and can be explored further by future work.

Acknowledgements

This work was supported in part by NSF grant DRL2112635. We are also thankful to the anonymous reviewers for their thoughtful suggestions.

Ethics and Broader Impact

Our synthetic datasets contain no linguistic or social information, and hence cannot introduce any type of social, gender and cultural biases in our analyses. The datasets used in the section 7 are publicly available, and should contribute towards the goal of reproducible research. In terms of broader impact, our results suggest that LMs accrue helpful inductive biases for non-linguistic reasoning during pretraining. This suggests that LMs can potentially be explored for a broader range of downstream applications rather than language-related tasks, which is the current predominant focus of these models. In the long run, making such foundational models available for learning a broad range of tasks from limited data can make predictive AI technologies more accessible than in the current day.

Limitations

In terms of findings, we find strong evidence of pretraining on text providing advantageous inductive biases for non-linguistic tasks. Our analysis in Section 6 suggests that this is not simply a regularization effect. However, it does not definitively rule out this possibility since direct comparisons between pretrained and non-pretrained networks (even of different sizes) are difficult. Also, the scope of our analysis here is limited to small to mid-sized language models (with tens of millions of parameters), rather than massive language models such as GPT3 (with tens of billions of parameters). Finally, we note that all tasks chosen for this analysis are formulated as classification, where the number of classes is not high. Hence, learning some of the tasks might easier than possible more general formulations. e.g., quantitative computation.

References

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.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Alon Brutzkus and Amir Globerson. 2021. On the inductive bias of a {cnn} for distributions with orthogonal patterns.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.

David P Helmbold and Philip M Long. 2015. On the inductive bias of dropout. *The Journal of Machine Learning Research*, 16(1):3403–3454.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre is not superb: Derivational morphology improves BERT's interpretation of complex words. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3594–3608, Online. Association for Computational Linguistics.

Valentin Hofmann, Janet B Pierrehumbert, and Hinrich Schütze. 2020. Dagobert: Generating derivational morphology with a pretrained language model. *arXiv preprint arXiv:2005.00672*.

Eugene Kharitonov and Rahma Chaabouni. 2021. What they do when in doubt: a study of inductive biases in seq2seq learners. In *International Conference on Learning Representations*.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside bert's linguistic knowledge. *arXiv preprint arXiv:1906.01698*.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2020. Does syntax need to grow on trees? sources of hierarchical inductive bias in sequence-to-sequence networks. *Transactions of the Association for Computational Linguistics*, 8:125–140.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, Valencia, Spain. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Carolyn Rose, and Eduard Hovy. 2019. Exploring numeracy in word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3374–3380, Florence, Italy. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Steven T Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic bulletin & review*, 21(5):1112–1130.

Luzi Sennhauser and Robert Berwick. 2018. Evaluating the ability of LSTMs to learn context-free grammars. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 115–124, Brussels, Belgium. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

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. 2019a. 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.

Mirac Suzgun, Yonatan Belinkov, and Stuart M. Shieber. 2019b. On evaluating the generalization of LSTM models in formal languages. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 277–286.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro Szekely. 2021. Representing numbers in NLP: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–656, Online. Association for Computational Linguistics.

Ivan Vulić, Simon Baker, Edoardo Maria Ponti, Ulla Petti, Ira Leviant, Kelly Wing, Olga Majewska, Eden Bar, Matt Malone, Thierry Poibeau, Roi Reichart, and Anna Korhonen. 2020a. Multi-SimLex: A large-scale evaluation of multilingual and crosslingual lexical semantic similarity. *Computational Linguistics*, 46(4):847–897.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020b. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7222–7240, Online. Association for Computational Linguistics.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. 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 5307–5315, Hong Kong, China. Association for Computational Linguistics.

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

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, Sheng-Fu Wang, Jason Phang, Anhad Mohananey, Phu Mon Htut, Paloma Jeretic, and Samuel R. Bowman. 2019. Investigating BERT's knowledge of language: Five analysis methods with NPIs. 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 2877–2887, Hong Kong, China. Association for Computational Linguistics.

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.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online. Association for Computational Linguistics.

Rui Xu, Xintao Wang, Kai Chen, Bolei Zhou, and Chen Change Loy. 2021. Positional encoding as spatial inductive bias in gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13569–13578.

Mengjie Zhao, Philipp Dufter, Yadollah Yaghoobzadeh, and Hinrich Schütze. 2020. Quantifying the contextualization of word representations with semantic class probing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1219–1234, Online. Association for Computational Linguistics.

A Appendix

Baseline	p-value
SNLI	5.45×10^{-5}
SNLI sort	3.33×10^{-4}
SNLI shuffle	5.5×10^{-4}
Amazon	7.48×10^{-5}
Amazon sort	7.2×10^{-5}
Amazon shuffle	4.5×10^{-5}
Multilingual BERT	9.07×10^{-4}
Chinese BERT	8.9×10^{-5}
Code BERT	8.1×10^{-5}
ROC	2.64×10^{-5}
Zipf distribution	7.45×10^{-5}
Uniform distribution	4.61×10^{-4}
Synthetic vocabulary	1.2×10^{-1}

Table 4: Statistical significance values (paired t-test) between non-pretrained model and other baseline BERT models trained on different datasets.

Baseline	p-value
SNLI	2.45×10^{-5}
SNLI sort	1.33×10^{-4}
SNLI shuffle	4.3×10^{-5}
Amazon	6.32×10^{-4}
Amazon sort	8.7×10^{-5}
Amazon shuffle	7.3×10^{-5}
Multilingual BERT	9.07×10^{-5}
ROC	2.14×10^{-3}
Zipf distribution	3.1×10^{-3}
Uniform distribution	4.61×10^{-4}
Synthetic vocabulary	1.3×10^{-1}

Table 5: Statistical significance values (paired t-test) between non-pretrained model and other baseline DeBERTA models trained on different datasets.

A.1 Implementation details

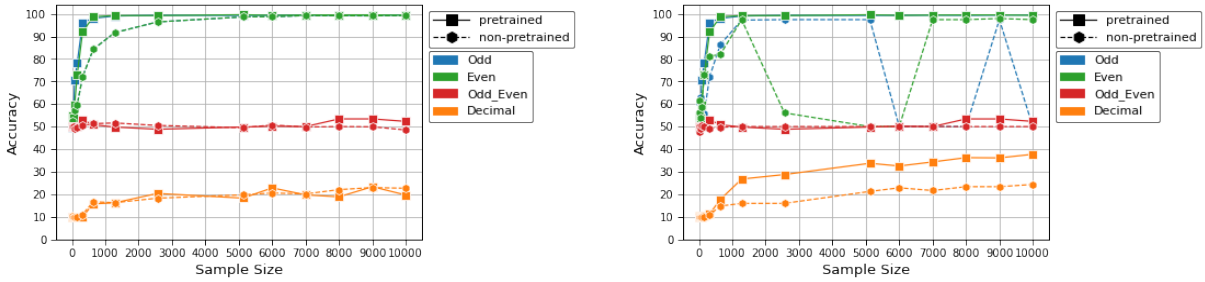
For transformer LMs, we add a fully connected classification layer on the top of final encoder layer. The pooled representations from the final encoder layer are then passed onto fully connected layer. We train these models in an end-to-end manner. For the RNN LMs, we first pretrain LM onto the task. The final word representations are the weighted sum of three layers. Max-pooling operation is applied on the time step dimension for these weighted representations. A final classification layer is trained with the pooled representations.

A.2 Computational requirements

All the models are run using PyTorch framework on 4 geforce gtx 1080 gpus. Each of the fine-tuning experiments takes about 5 gpu hours and pre-training takes about 10 gpu hours.

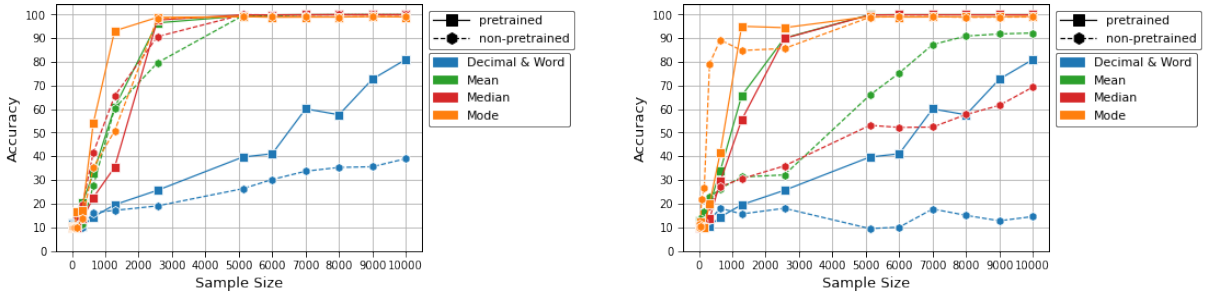
A.3 Statistical significance

We perform a paired t-test between pretrained and non-pretrained models of the LMs on all the tasks. The statistical significance values are shown in the table 6. We also calculated the paired t-value between non-pretrained model and BERT and DeBERTA pretrained on different datasets. The paired t-values are shown in the table 4 and 5.



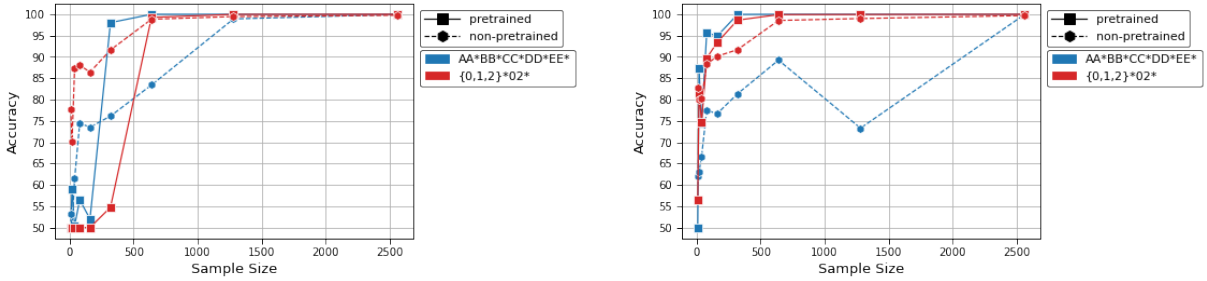
(a) DeBERTa (b) BERT large

Figure A.1: Performance comparison of pretrained and non-pretrained models of DeBERTa and BERT large on four quantitative computation tasks (odd classification, even classification, odd even classification and decimal operation).



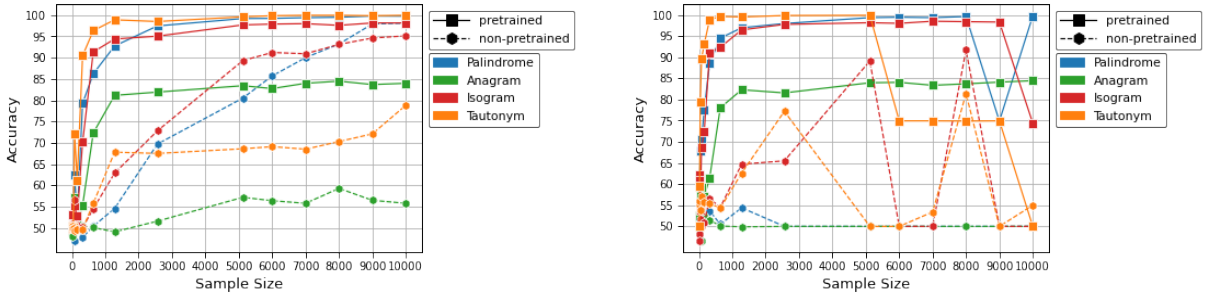
(a) DeBERTa (b) BERT large

Figure A.2: Performance comparison of pretrained and non-pretrained models of DeBERTa and BERT large on four quantitative tasks (mean, median, mode, decimal & word operation).



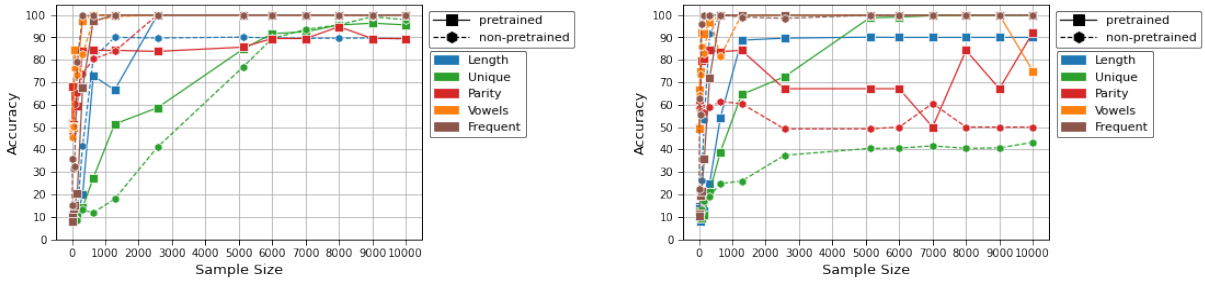
(a) BERT small (b) ELMO

Figure A.3: Performance comparison of pretrained and non-pretrained models of DeBERTa, and BERT large on regular expression tasks (AA*BB*CC*DD*EE* and recognize {0,1,2}*02*).



(a) DeBERTa (b) BERT large

Figure A.4: Performance comparison of pretrained and non-pretrained models of DeBERTa and BERT large on four string reasoning (palindrome, anagram, isogram and tautonym classification).



(a) DeBERTa

(b) BERT large

Figure A.5: Performance comparison of pretrained and non-pretrained models of DeBERTa and BERT large on five string reasoning tasks (length of a string, maximum frequent character, vowels classification, parity check and count of unique character).

Task	BERT small	DeBERTa	BERT large	ELMO
Odd classification	10.4×10^{-2}	8.8×10^{-1}	2.9×10^{-3}	7.35×10^{-7}
Even classification	8.1×10^{-2}	8.7×10^{-2}	5.25×10^{-3}	7.35×10^{-7}
Odd even classification	2.2×10^{-1}	6.96×10^{-7}	6.46×10^{-4}	7.35×10^{-7}
Decimal operation	4.1×10^{-4}	7.07×10^{-1}	1.35×10^{-5}	3.49×10^{-7}
Decimal & word operation	6.85×10^{-8}	6.43×10^{-7}	4.34×10^{-8}	5.39×10^{-7}
Mean	9.5×10^{-2}	7.56×10^{-1}	7.8×10^{-6}	2.2×10^{-7}
Median	9.28×10^{-6}	8.04×10^{-1}	5.68×10^{-7}	1.99×10^{-7}
Mode	9.2×10^{-2}	2.27×10^{-1}	9.2×10^{-1}	3.35×10^{-7}
Recognize {0,1,2}*02*	1.31×10^{-1}	8.4×10^{-1}	4.34×10^{-1}	5.48×10^{-5}
Recognize AA*BB*CC*DD*EE*	4.06×10^{-1}	6.97×10^{-1}	4.02×10^{-1}	2.39×10^{-6}
Palindrome classification	4.34×10^{-7}	2.1×10^{-3}	1.85×10^{-7}	1.97×10^{-6}
Anagram classification	5.1×10^{-6}	1.44×10^{-6}	3.45×10^{-7}	7.46×10^{-6}
Isogram classification	1.28×10^{-7}	4.77×10^{-3}	3.47×10^{-4}	2.18×10^{-6}
Tautonym classification	1.92×10^{-7}	1.29×10^{-5}	1.69×10^{-8}	4.39×10^{-6}
Length of a string	2.7×10^{-1}	1.27×10^{-4}	3.39×10^{-4}	7.07×10^{-4}
Count of unique characters	1.79×10^{-4}	2.7×10^{-2}	1.23×10^{-7}	3.18×10^{-6}
Parity check	2.68×10^{-4}	4.66×10^{-4}	4.34×10^{-7}	6.05×10^{-6}
Vowels classification	4.26×10^{-1}	9.5×10^{-1}	7.22×10^{-1}	5.11×10^{-2}
Maximum frequent character	5.02×10^{-1}	5.65×10^{-1}	6.07×10^{-1}	6.47×10^{-1}

Table 6: Statistical significance values (paired t-test) between pretrained and non-pretrained model on all the tasks.