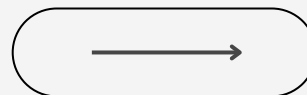


DATE
13/03/2025



BETWEEN CIRCUITS AND CHOMSKY: PRE-PRETRAINING ON FORMAL LANGUAGES IMPARTS LINGUISTIC BIASES

Michael Y. Hu et al., 2025 (New York University)



UNIVERSITY OF GRONINGEN
InCLoW Reading Group

PRESENTED BY
Francesca Padovani

01 INTRODUCTORY CONTEXT

→ TRANSFORMERS' DATA HUNGRYNESS

→ PRE-PRETRAINING ON FORMAL LANGUAGE (before training on natural language)

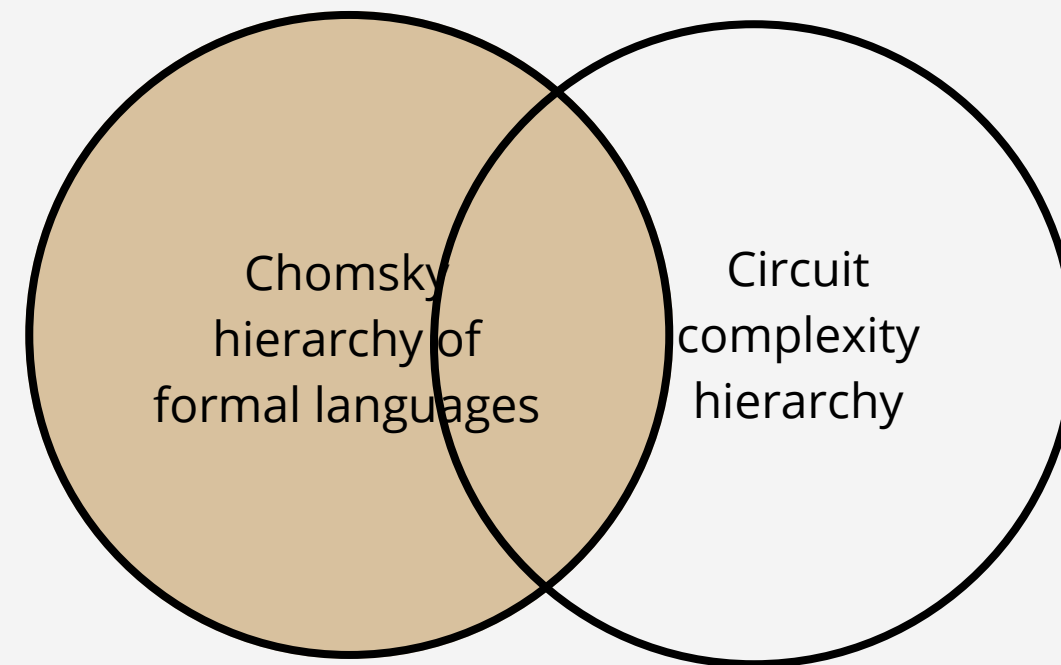
→ **CONTEXT SENSITIVE LANGUAGES** transfer best to natural language learning, but **Transformers struggle** with some context-sensitive languages (they can't always learn next-token prediction for them)

How positive transfer occurs, if a generalizing solution cannot always be learned?

Which characteristics of formal languages make them effective for pre-pretraining?

01 MAIN HYPOTESIS

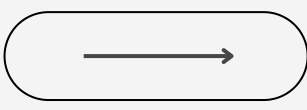
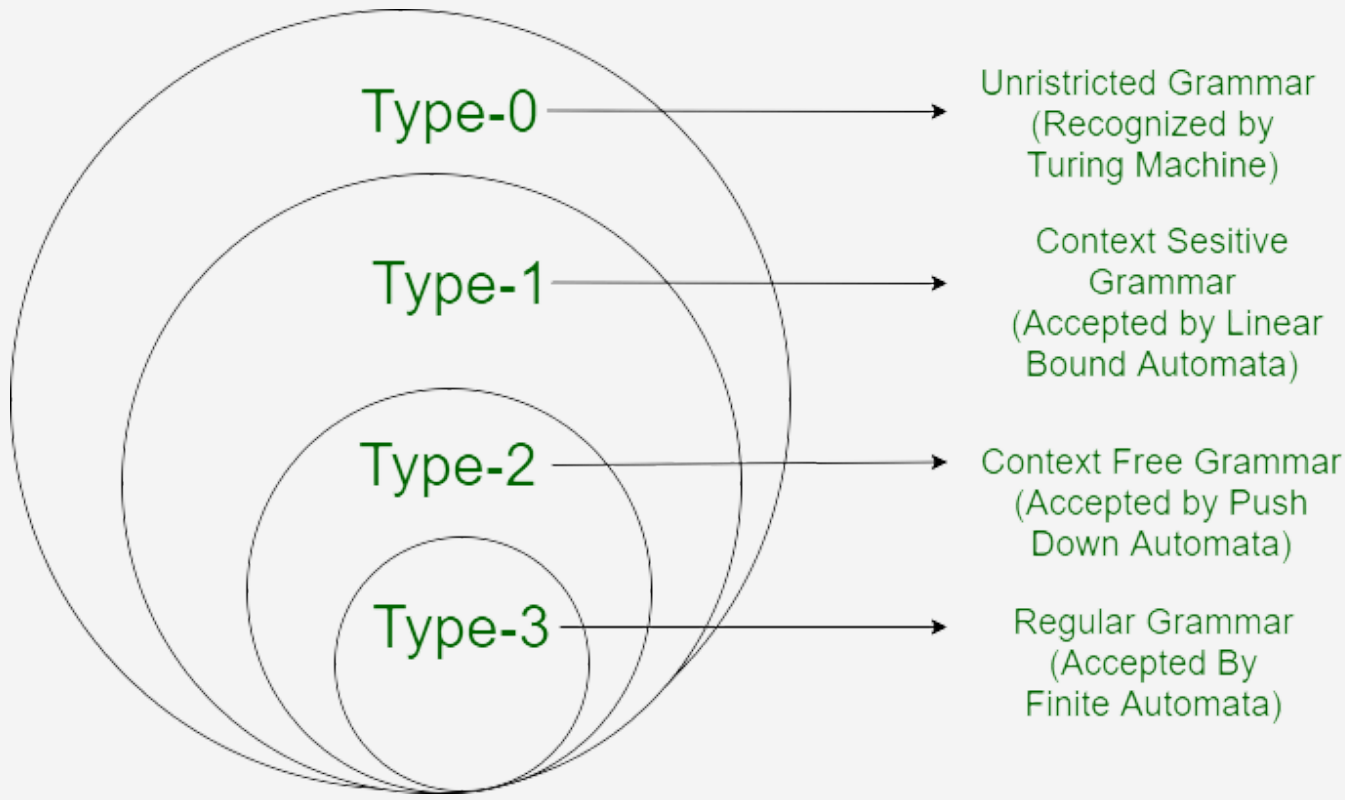
Optimal transfer from formal to natural language occurs at the intersection of two theoretical hierarchies



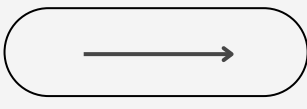
Specifically they hypothesize that effective pre-pretraining languages should be:

1. *expressive enough to capture hierarchical natural language dependencies* (**focus on the LANGUAGE**)
2. *learnable by transformers in a length-generalizing way* (**focus on the MODEL**)

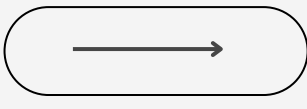
THE CHOMSKY HIERARCHY



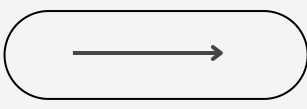
Only a select few phenomena in natural language require context sensitivity: cross-serial dependencies in Swiss German or anaphora



The rest can be modeled using context-free grammars

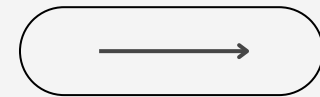


A classic context-free language is *k*-**Dyck** - the language of well-balanced parentheses with *k* bracket types ex. $([])[[]]$

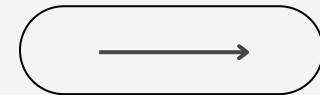


Removing the constraint that Dyck braces must be well-nested (but enforcing that every opening brace must be closed and vice versa) yields *k*-**Shuffle Dyck** (strictly context-sensitive rather than context-free)

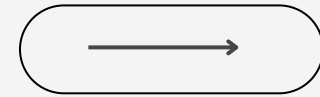
THE CIRCUIT COMPLEXITY HIERARCHY



FOCUS OF THIS WORK: TRANSFORMERS



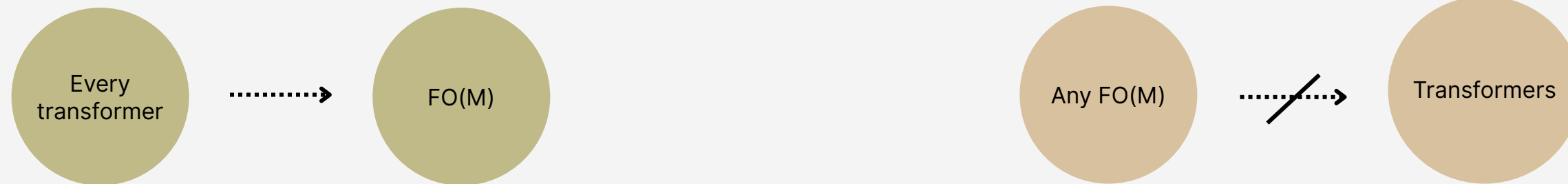
WHICH IS THE EXPRESSIVE POWER OF TRANSFORMERS?



TWO LOGICS THAT EMERGE FROM THIS CIRCUIT COMPLEXITY VIEWPOINT: **FO(M), C-RASP**

First-order logic with majority

well-defined variant of Restricted Access Sequence Processing programming lang



- FO(M) = any language that a transformer can learn can also be described using a FO(M) program
- C-RASP = is a restriction of FO(M) designed to be a lower bound on what transformers can express

02

C-RASP and experimental motivation

- **C-RASP similar to FO(M), but with some restrictions** = crucially in C-RASP each predicate can only refer to one index variable i , whereas in FO(M), predicates can refer to two (or more) indices i, j introduced by different quantifiers.
- **Recent works (Zhou et al., 2024; Huang et al., 2025)** = show connection between C-RASP and length generalization of transformers. The definability of a language L in C-RASP predicts whether transformers can reliably length-generalize when trained on strings from L .
- **INTERPRETATION** = One interpretation of this finding is that mechanisms expressible in C-RASP may be more robustly learnable for transformers.
- **HYPOTHESIS** = it is better to pre-pretrain on formal languages that can be defined in C-RASP so that the learned representations can transfer reliably.

METHODS

DEFINING PRE-PRETRAINING

Use optimizer $A(D, t, \theta_{init})$ which returns parameters θ_t after t timesteps. They apply A sequentially:

1. **Pre-pretrain** for t_0 steps on dataset D_{ppt} to obtain model parameters θ_{t_0}
2. **Pretrain** for t_1 steps on dataset D_{pt} to obtain θ_{t_1}

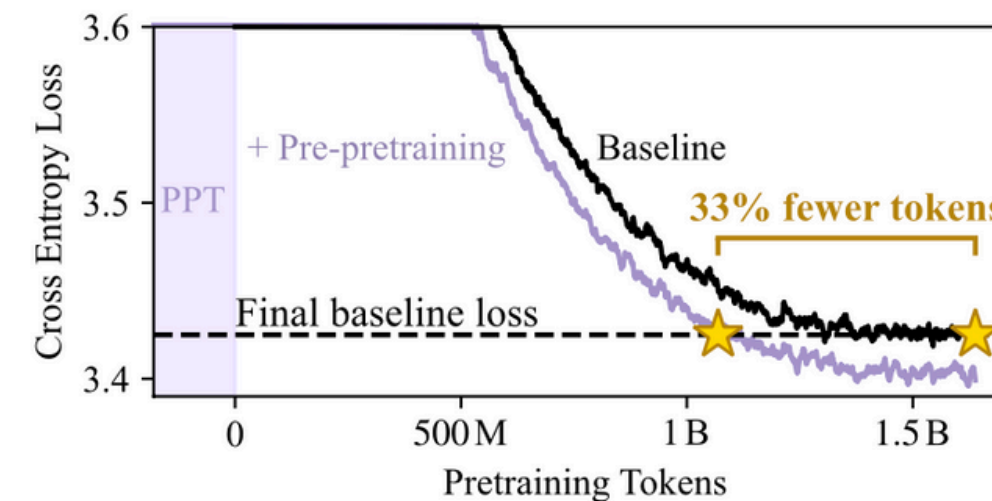
Their objective is to minimize the expected loss on the pretraining dataset:
 $\text{argmin}_{\theta_1} E[l(D_{pt}, \theta_{t_1})]$.

KEY CONSTRAINT = keep everything else fixed (optimizer, t_1 , and D_{pt}), so the only variables are:

- Pre-pretraining dataset (D_{ppt})
- Pre-pretraining duration (t_0)

This way, they can directly test whether pre-pretraining on formal languages is beneficial.

	Context-free	Context-sensitive
C-RASP	1-Dyck	<i>k</i> -Shuffle Dyck
FO(M)	<i>k</i> -Dyck	<i>ww</i>



BASELINES OF COMPARISON

- No pre-pretraining ($t_0=0$)
- Random binary strings ("0110101011100010")
- Random strings of k integers ("3 7 2 5 8 1 0 4 6 9")
- Held-out natural language data from the same distribution as D_{pt}

METHODS

EVALUATION

validation loss (0), **BLiMP** (1) and **verbatim retrieval** (2)

0. **Lower validation loss** compared to the no pre-pretraining baseline would indicate that pre- pretraining on formal languages is beneficial

1. **Accuracy** on BLiMP as proportion of examples where the grammatical sentence is assigned higher likelihood than the ungrammatical one (Warstadt et al., 2020a)

2. **Verbatim retrieval** tests LMs on text passages with repeated lists (Armeni et al., 2022, 2024); the model is expected to assign a very high likelihood to the words in the second repetition of the list

"Before the meeting, Mary wrote down the following list of words: window, door, roof. After the meeting, she took a break and had a cup of coffee. When she got back, she read the list again: window, door, roof."

METHODS

EFFICIENCY

Calculate the **marginal rate of substitution (MRS)** between formal and natural language at 10,000 steps of natural language pretraining

In other words, if we train on 500 steps of the formal language, how many more steps does it take for the natural language-only baseline to catch up?

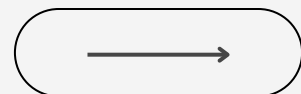
Suppose we have two scenarios that achieve the same loss:

- No pre-pretraining → Takes (0,10000) steps of natural language training
- With pre-pretraining (500 steps on formal language) → Takes (500,6000) steps of natural language training

$$1 - \frac{6,000}{10,000} + \frac{500}{10,000} = 35\%$$

pre-pretraining on formal languages reduces the number of required natural language training steps by 35%

good pre-pretraining language



minimize amount of pretraining steps + improve evaluation performance

EXPRESSIVITY HYPOTHESIS

A formal language that confers a helpful inductive bias should be hierarchically structured (either context-free or context-sensitive) and definable in C-RASP.

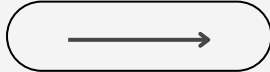
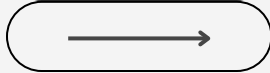
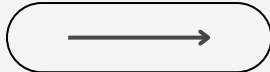
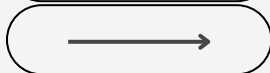
Language	Example		
1-Dyck	((()))		the nested parentheses language. Context-free, in C-RASP
k -Dyck	([{ }])		contains k different types of parentheses. Context-free, in FO(M)\C-RASP
k -Shuffle Dyck	([{]) }		k -Dyck with cross-serial dependencies. Context-sensitive, in C-RASP
ww	1 2 3 1 2 3		the copy language. Context-sensitive, in FO(M)\C-RASP

Table 1: Examples of our pre-pretraining languages.

To reduce confounds: they build 1-Dyck, k -Dyck, and k -Shuffle Dyck **corpora with matching depth distributions**.

During corpus generation - randomly opening or closing parentheses with probability $p = 0.5$ - which yields a harmonic distribution over depths.

All language models are pre-pretrained on the same number of tokens with sequence packing.

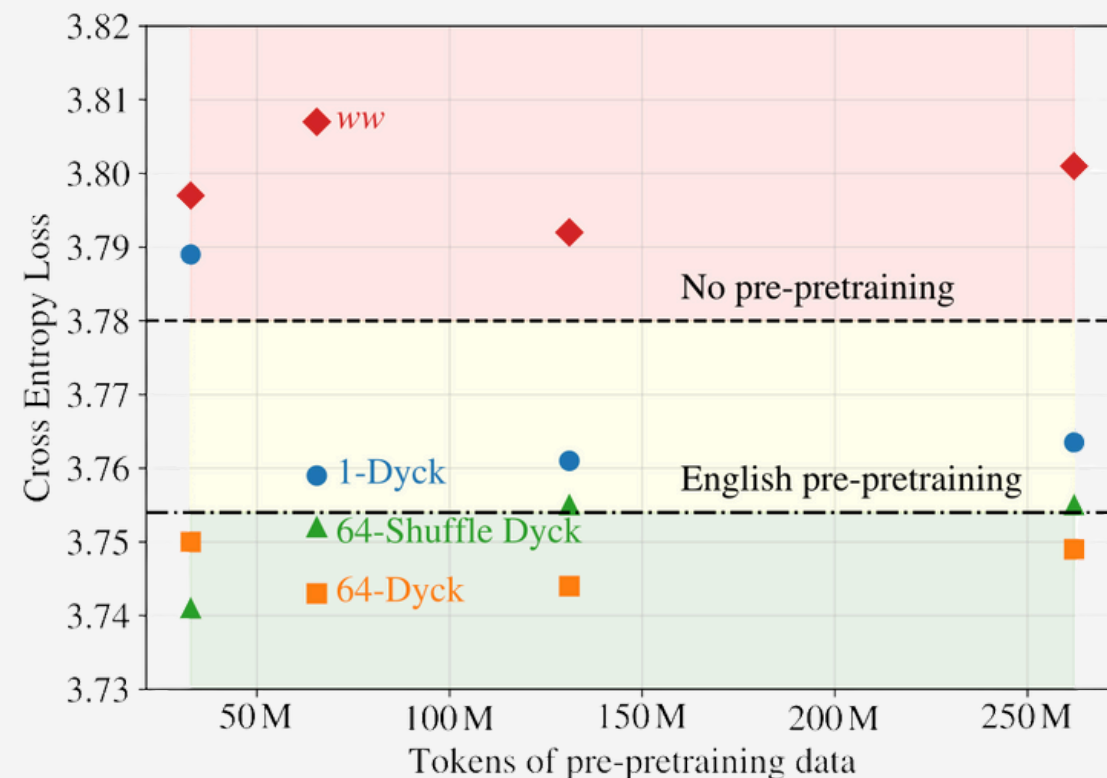
TRAINING

- trained Pythia 160M models (Biderman et al., 2023)
- for 10,000 steps
- on roughly 665 million tokens
- C4 as the natural language dataset (Raffel et al., 2019)

For natural language...

OPTIMAL PRE-PRETRAINING DURATION VARIES BY LANGUAGE

- Different formal languages require different amounts of pre-pretraining for best transfer to natural language
- The study tests four durations (from 30M to 260M tokens, 500 to 4000 gradient updates)



- Most efficient formal language is k -Shuffle Dyck, with $t_0^* = 500$.
- k -Dyck also outperform natural language pre-pretraining, but optimum is achieved around $t_0^* = 1000$.

EVALUATION

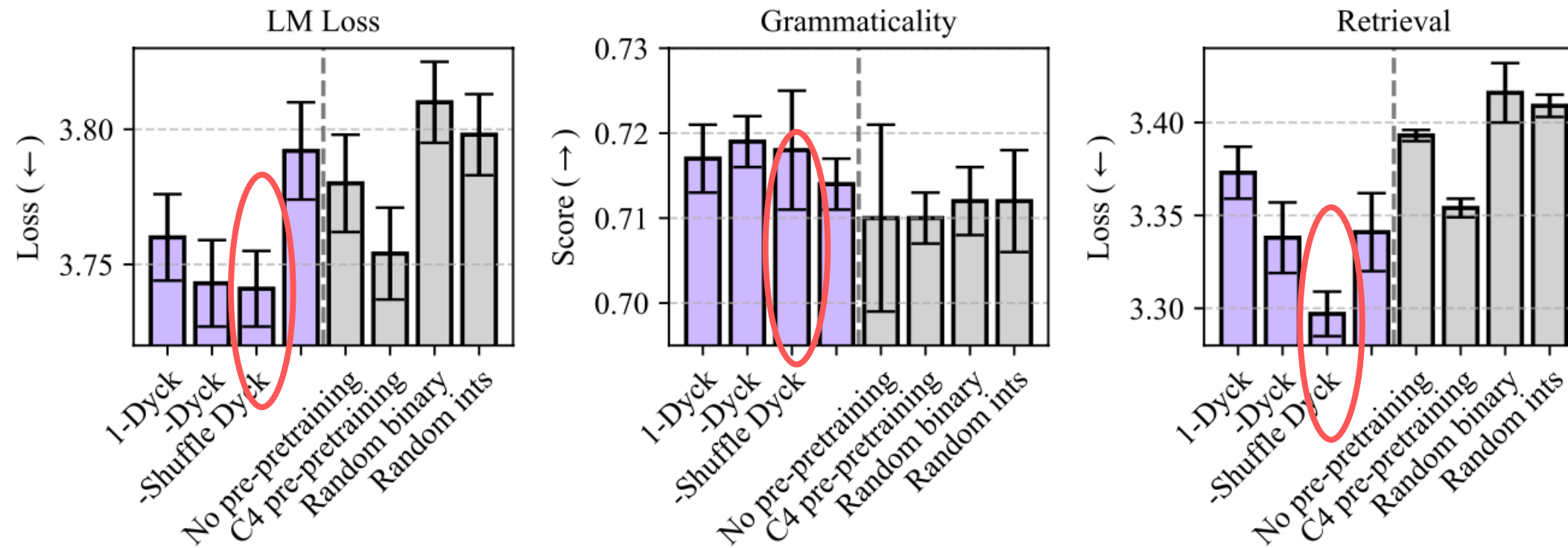
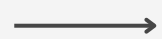


Figure 2: Evaluating models at the optimal amount of pre-pretraining t_0^* for each formal language. k -Shuffle Dyck performs the best overall on our evaluation metrics.



k -Shuffle Dyck is the best-performing formal language on the validation set of C4, followed by k -Dyck



pre-training on all four formal languages improves accuracy in grammaticality, but pre-training on natural language (C4) does not [pre-training induces representations useful for modeling hierarchical structure]



pre-training on either random binary strings or k -integer strings has a negative effect: it results in higher validation loss than no pre-training

PRUNING

What is the mechanism by which pre-pretraining facilitates the learning of natural language?

SUBNETWORK HYPOTHESIS

- Pre-pretraining activates a subnetwork (MMM) in the model
- This subnetwork persists and improves natural language learning

METHODS

Step 1: Pre-train on formal language dataset (D_{ppt})

Step 2: Prune attention heads using core pruning algorithm, Bhaskar et al. (2024)

Step 3: Fine-tune on natural language (D_{pt})

Step 4: Compare pruned subnetwork (MMM) vs. randomly pruned model (M_{null})

EXPECTED RESULTS

- If no impact: $MMM \approx M_{null}$
- If positive transfer: MMM performs significantly better

PRUNING (RESULTS)

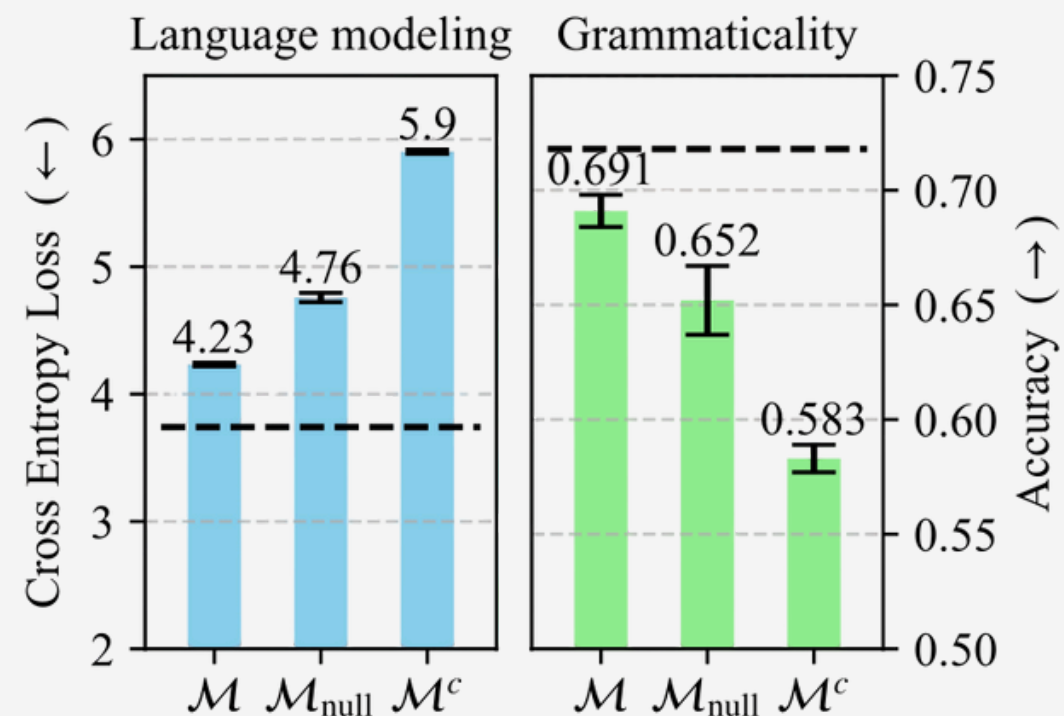


Figure 4: Language modeling and grammaticality performance for the learned subnetwork \mathcal{M} , its complement \mathcal{M}^c , and randomly sampled masks $\mathcal{M}_{\text{null}}$. \mathcal{M} outperforms \mathcal{M}^c and $\mathcal{M}_{\text{null}}$, indicating that the subnetwork learned during pre-pretraining continues to play a critical role after training on natural language. Dashed lines indicate performance of the base model without pruning.

- \mathcal{M} (structured pruning – learned subnetwork)
- $\mathcal{M}_{\text{null}}$ (random pruning – control group)
- \mathcal{M}^c (complement of \mathcal{M} – the pruned part of the model)

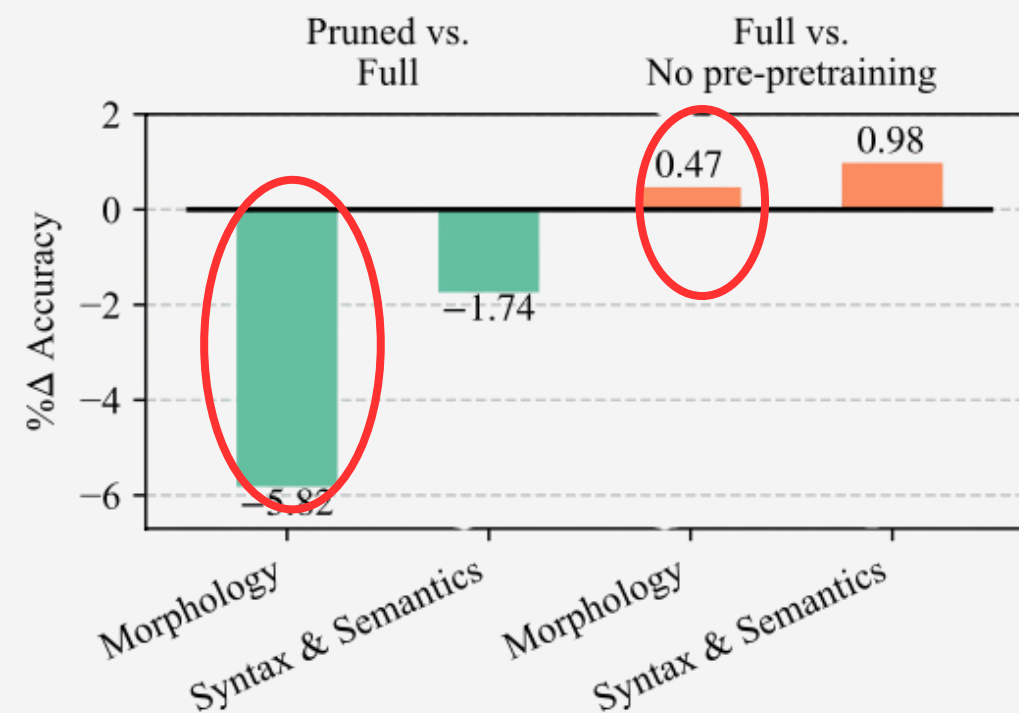
They find that \mathcal{M} outperforms random masking and \mathcal{M}^c . They show that \mathcal{M} does not achieve the performance of the full network, indicating that attention heads outside of \mathcal{M} are also useful for natural language.

SYNTACTIC SUBNETWORK

Subnetwork **M** mostly **implements syntax**, but pre-pretraining **also helps** models learn non-syntactic knowledge like **morphology**, or **word structure**.

TO OBSERVE THIS

They compare M's performance on BLiMP against that of the full network, aggregated by the classification of linguistic phenomena as reflecting morphology (27% of BLiMP examples) versus syntax and semantics (73%).

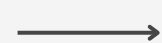


CONCLUSION

pre-pretraining has second-order effects: in addition to syntactic benefits, pre-pretraining can improve performance on non-syntactic tasks.

ANALYZING SHUFFLE DYCK

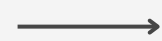
Does the success of k -Shuffle Dyck come from its hierarchical structure, or could the model learn similar patterns from simpler statistical features?



Neural networks have a **distributional simplicity bias** (DSB) (Belrose et al., 2024) they learn simpler statistical patterns, such as the mean and covariance of their representations, before progressing to higher-order relationships



Removing the rule-based structure of k -Shuffle Dyck while preserving statistical properties still leads to improvements?

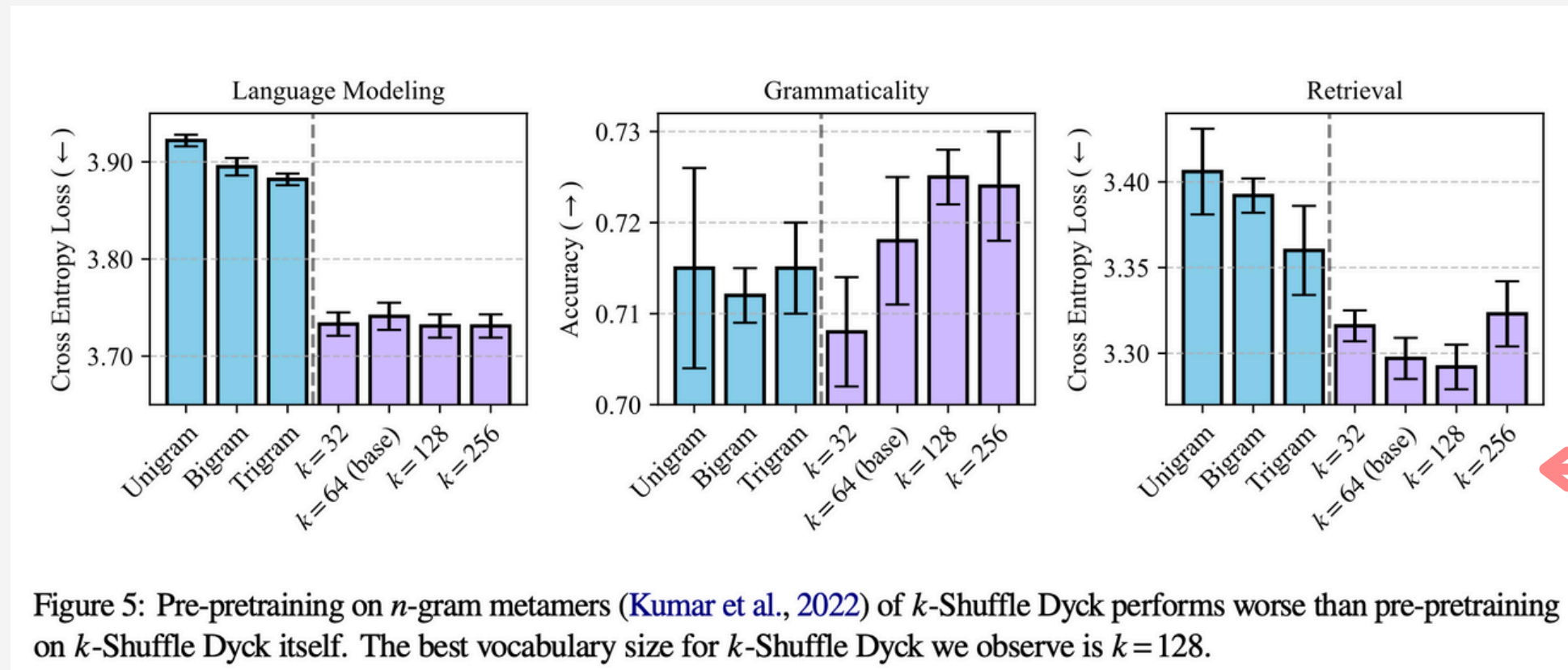


Ablation of Rule-Based Structure: Instead of training on true k -Shuffle Dyck sequences, **they generate synthetic datasets (metamer datasets)** that match k -Shuffle Dyck in certain statistical properties but lack its formal structure.



1. train unigram, bigram, and trigram models on the k -Shuffle Dyck corpus
2. use these models to generate new datasets that mimic statistical properties but do not follow the rule-based syntax of Dyck languages

ANALYZING SHUFFLE DYCK



They also test different k for k -shuffle Dyck

LARGER SCALE

examine whether their results generalize to larger settings by training Pythia-1B for 1.63B tokens on C4 (25,000 steps)

In this setting, pre-pretraining on k -Shuffle Dyck continues to outperform on all evaluation metrics and achieves the final loss of no pre-pretraining in 1.10B total tokens. This equates to a token **efficiency gain of 33%** □
Pre-pretraining could increase the efficiency of large-scale pretraining as well.

CONCLUSIONS AND REMARKS

→ BLOCK TRAINING : first on formal languages and then on natural language
MIXING TRAINING: mixing formal and natural language during training could lead to better performance (Korbak et al., 2023)

→ They evaluated efficiency in a setting where pretraining data is plentiful, and one can train without running several epochs over the same data

→ A low-resource setting INTERESTING and may yield **different scaling properties with respect to pre-pretraining data** (Muennighoff et al., 2023). Relevant for non-English languages

→ Finally, this work only considers transformers. Circuit complexity has also quantified the expressive power of neural networks like RNNs and state-space models (Merrill et al., 2020, 2024), and it would be interesting to extend our results to these architectures.