Priming: Multi-Stage Pretraining using Formal Languages with Ascending Complexity

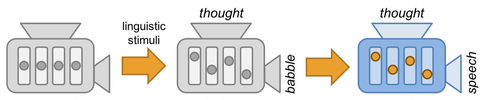
Tianyi Niu

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

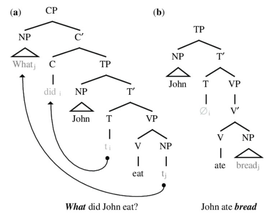
Background

Inductive Biases in Human Language

- Humans learn language with *less data and parameters* than Language Models.
- Many linguistics argued for the (controversial) case of *innate grammar*.
- Natural languages (can be interpreted to) have inherent structure, called a *parse tree*

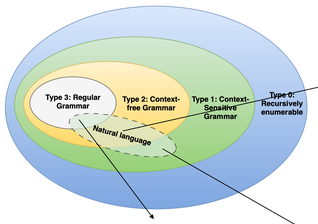


(Boonkwan, 2018)



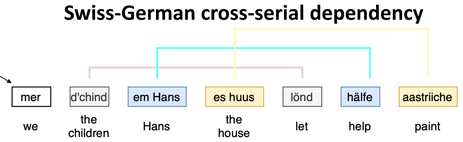
(Derrick & Archambault, 2009)

Formal Grammars & The Chomsky Hierarchy



English Center Embedding

Swiss-German cross-serial dependency



(Shieber, 1985)

Ponapean Reduplication

Part of the stem is copied and attached to the stem (Rehg, 1981)

duhp "dive" **duduhp** "be diving"
mihk "suck" **mimihk** "be sucking"

Methodology

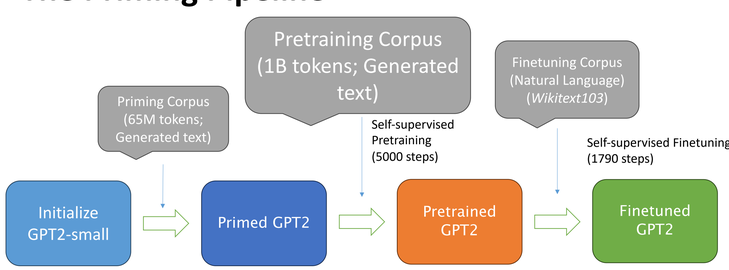
Research Questions:

- Can priming lead to stronger biases (a.k.a lower perplexity)?
- Can priming increase training efficiency?

Regular → Context-Free → Context-Sensitive

Priming | Pretraining | Finetuning

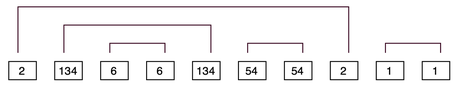
The Priming Pipeline



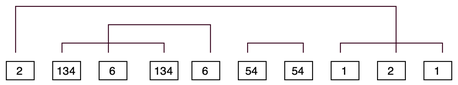
Datasets

Pretraining Datasets

- NEST (Context-Free)



- FLAT (Context-Sensitive)




```
unclosed_stack = stack()
sequence = []
while num_tokens < 65M:
    open = Bernoulli(0.49)
    if open == 1:
        vocab_token = uniform(1, 500)
        sequence.append(vocab_token)
        unclosed_stack.add(vocab_token)
    if open == 0:
        vocab_token = unclosed_stack.pop()
        sequence.append(vocab_token)
```

```
unclosed_list = list()
sequence = []
while num_tokens < 65M:
    open = Bernoulli(0.49)
    if open == 1:
        vocab_token = uniform(1, 500)
        sequence.append(vocab_token)
        unclosed_list.add(vocab_token)
    if open == 0:
        vocab_token = shuffle(unclosed_stack).pop()
        sequence.append(vocab_token)
```

Priming Datasets

Regular

TOM7: $0^+1^+0^+1^+$

TOM7-5(type):
$0^+1^+2^+3^+4^+5^+0^+1^+2^+3^+4^+5^+$

TOM7-10(type):
$0^+1^+...9^+10^+0^+1^+...9^+10^+$

TOM2: (01)+

TOM5:
Even number of 0's and 1's

Context-Free

NEST-MINI:
Equivalent to NEST, reduced to 65M tokens

PALINDROME:
$S \rightarrow \epsilon|1S1|0S0|$

Baselines

RANDOM **NO-PRIMING**

Context-Sensitive

FLAT-MAXARC3:
Same algorithm as FLAT, limit maximum dependency arc length to 3

FLAT-MAXARC5:
Same algorithm as FLAT, limit maximum dependency arc length to 5

BACH: $0^n1^n2^n3^n$

Example Sequence Generation

1. Randomly select three indices $idx1$ $idx2$ $idx3$
2. Map type tokens (0, 1) to vocabulary tokens (0, 1, 2, …, 499)

_ _ _ _ _ _ _ … _ _ _ _ _ _ _ _

512

0, 0, 0, 0, 1, 1, …, 1, 0, 0, 1, 1, 1

[0:idx1] [Idx1:idx2] [Idx2:idx3] [Idx3:512]

123, 31, 13, 31, 386, 455, …, 2298, 21, 203, 400, 257, 289

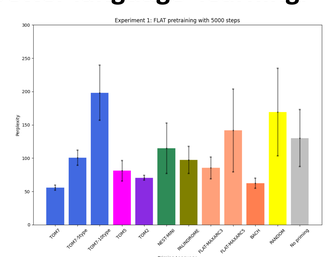
TOM7: $0^+1^+0^+1^+$

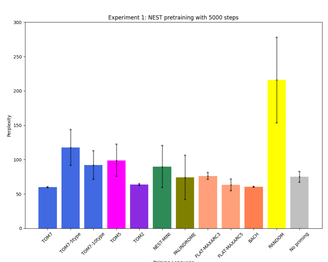
assignments = {
"0": [0,1,…,249],
"1": [250, 251, …, 499]
}

probabilities = {
"0": [1/250, 1/250, …, 1/250],
"1": [1/250, 1/250, …/ 1/250]
}

Experiment 1

Objective: Investigate whether the priming step leads to better language learning.





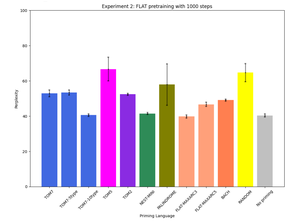
Procedure:
(1) Prime for 500 steps
(2) Pretrain for **5000** steps,
(3) Finetune on Wikitext103.
(4) Evaluate perplexity
(lower the better)

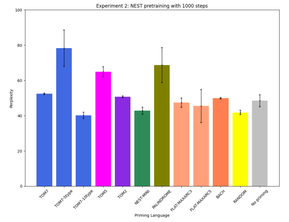
Takeaways:

1. Low PPL, low variance. High PPL, high variance.
2. FLAT less tolerant to more types and longer dependency arcs compared to NEST
3. No strong correlation between complexity and performance
4. Simple languages with few types and repetitive patterns are the most performant.
5. One-to-one dependencies deteriorate performance

Experiment 2

Objective: Investigate whether priming allows for more efficient pretraining.





Procedure:
(1) Prime for 500 steps
(2) Pretrain for **1000** steps,
(3) Finetune on Wikitext103.
(4) Evaluate perplexity
(lower the better)

Takeaways:

1. Less pretraining steps results in lower perplexity and lower variance for all languages
2. Priming no longer outperforms baselines
3. Patterns found in Experiment 1 are no longer present
4. Pretrain T2T variants outperforms simple languages

Conclusions

1. Priming allows for better language learning? **Yes and No**

Yes – When pretrained for **a sufficient number of steps**, priming on non-linguistic datasets generated from simple, repetitive patterns **substantially improve performance.**

No – When pretrained for a few steps, priming does not offer substantial gains over baselines. In fact, simple, repetitive patterns **deteriorate performance**.

2. Priming enable more efficient training? **Maybe**

Maybe - Perplexity on all language improve when pretrained for less steps, *including no priming baseline.*

Maybe – Checkpoints only taken at 1k and 5k steps & 1k may be overfitting FT dataset

Discussion

Why and what is "sufficient number of steps?"

A potential explanation: spurious correlations between location and value:

0, 0, 0, 0, 1, 1, …, 1, 0, 0, 1, 1, 1

[0:idx1] [Idx1:idx2] [Idx2:idx3] [Idx3:512]

123, 31, 13, 31, 386, 455, …, 228, 21, 203, 400, 257, 289

assignments = {
"0": [0,1,…,249],
"1": [250, 251, …, 499]
}

probabilities = {
"0": [1/250, 1/250, …, 1/250],
"1": [1/250, 1/250, …/ 1/250]
}

In experiment 2 (pretrain 1k steps):

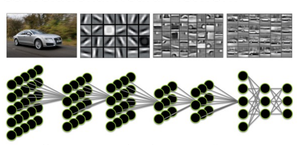
- Simple languages: the model incorrectly learns the spurious correlation between location and token value.
- T2T languages: the model is primed then pretrained on the same (or similar) datasets, essentially adding 500 extra training steps

In experiment 1 (pretrain 5k steps):

- Simple languages: despite learning spurious correlations, the model later corrects itself and "overwrites" the spurious correlation, and learns more abstract patterns, leading to better generalization
- T2T languages: despite extra training steps, the model does not generalize as well.

Potential Parallels Between Vision and Text

Convolutional Neural Network



https://blogs.nvidia.com/blog/whats-the-difference-between-a-cnn-and-an-rnn/

Transformer LM

Word types: [noun][verb][noun] → Lower layers

Phrasal Structure: $_{TP}$[noun $_{VP}$[verb $_{DP}$[noun]]] → Middle layers

Token-to-token dependencies: He [ate […]] → Higher layers