How Does Code Pretraining Affect Language Model Task Performance?

Jackson Petty[1*], Sjoerd van Steenkiste[2], and Tal Linzen[2]

* *Work done as a Student Researcher at Google Research*

1 NYU 2 Google Research

Question: Does pretraining on *source code* help LLMs make more compositional generalizations?

1. What is compositional generalization?

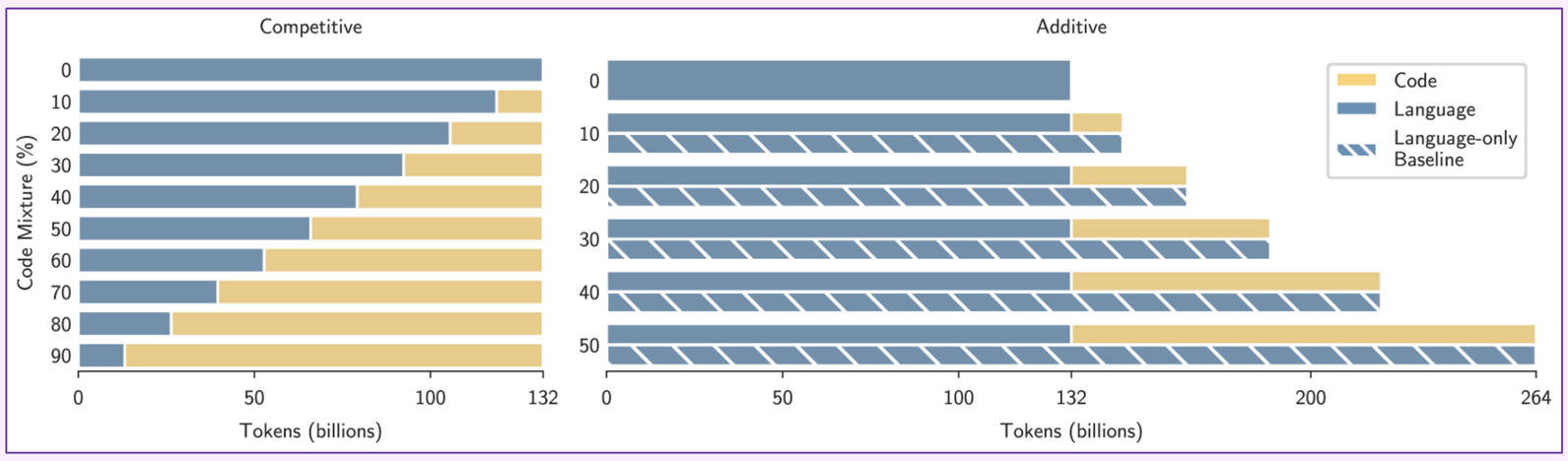
Generalize from **known pieces** to **(infinite) novel, well-formed combinations**

Training input (hedgehog is subject)		*Output*
`the hedgehog ate the cake`		*eat*(***agent=hedgehog***, *theme=cake*)
`the hedgehog saw a child`		*see*(***agent=hedgehog***, *theme=child*)
`hedgehogs swim`	→	*swim*(***agent=hedgehog***)
Generalization (hedgehog is object)		
`the boy loves the hedgehog`		*love*(*agent=boy*, ***theme=hedgehog***)

2. Evaluating compositional generalization

Dataset	*Examples*
COGS	A hedgehog ate the cake . *cake(x4); hedgehog(x1) and eat.agent(x2, x1) and eat.theme(x2, x4)
COGS-vf	A hedgehog ate the cake on the bed . eat(agent = hedgehog,theme =*cake(nmod.on =*bed))
English Passivization	our vultures admired her walrus above some zebra . her walrus above some zebra was admired by our vultures .

3. Experimental setup

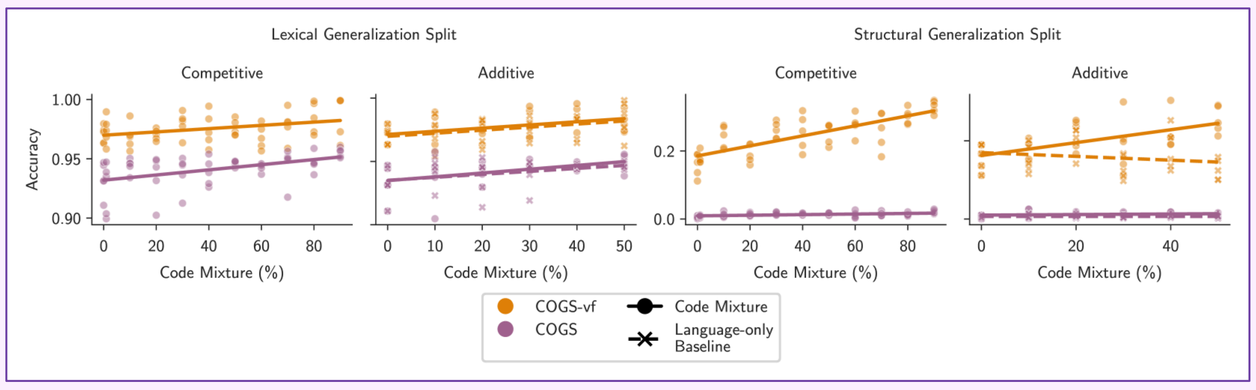


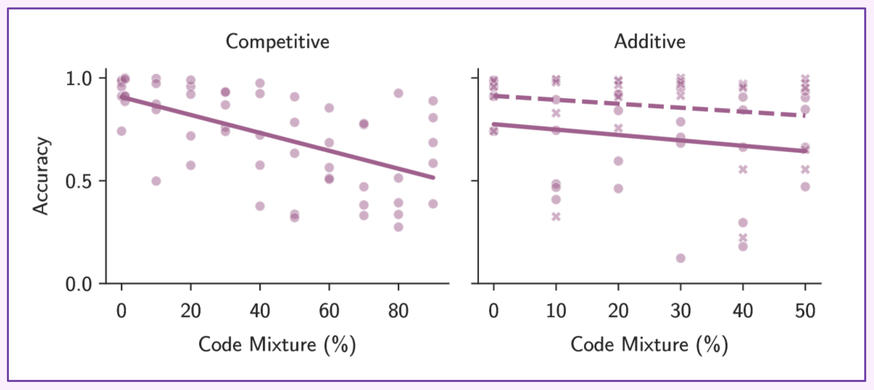
1. Pretrain 400M-parameter LLMs on mixtures of code (GitHub) and language (C4)
2. Finetune on training splits of compositional generalization datasets
3. Evaluate on generalization split

4. Results

Code helps on COGS & COGS-vf (semantic parsing)

but hurts on purely natural-language tasks, like English Passivization





Answer: Yes, depending on the format

Code can help models generalize more compositionally, but only in cases where the output domain has formal structure