Auto Regressive Next Token Predictors are Universal Learners

Jacopo Peroni, Aryan Rahbari

Auto-regressive Learning

An auto-regressive next-token predictor has the following form: $h \in H = H_1 \times \cdots \times H_T$

$h^{(1)}(x) = h(x, \emptyset) \qquad x \in \mathbb{D}^n$

$h^{(2)}(x) = h(x, (h^{(1)}(x)))$

$h^{(T)}(x) = h(x, (h^{(1)}(x), \ldots, h^{(T-1)}(x)))$

We say that a function $f : \mathbb{D}^n \longrightarrow \mathbb{D}$ is computed by h if $h^{(T)}(x) = f(x)$

Trained with chain of thought s.t. next token prediction becomes learnable for H

Linear Decoders

A particular example of AR models are linear decoders defined as follows

$$h_W(x, z) = argmax_{D \in \mathbb{D}} \langle W_D, \psi(x, z) \rangle$$

Where the functional space is

$$H^{lin} = H_1^{lin} \times \cdots \times H_T^{lin} \qquad H_t^{lin} = \{h_W | W \in \mathbb{R}^{\mathbb{D} \times d \times (n+t)}\}$$

Theorem: Every Turing computable function is AR computable with respect to H^{lin}

Length Complexity

Given a function $f : \mathbb{D}^n \longrightarrow \mathbb{D}$ we say h computes f with length complexity T if $h^{(T)}(x) = f(x)$

Theorem: Any $f : \mathbb{D}^n \longrightarrow \mathbb{D}$ that is Turing computable in time T(n) and any distribution $\hat{D}$ over inputs of size n, there is a dataset of strings of tokens of length poly(T(n)), such that training a linear AR model over this dataset leads to a function that computes f

Trade off between computational and length complexity

Experiment

Prompt:	1394×8618=
Output (MLP):	(4×1+9×10+3×100+1×1000)× (8×1+1×10+6×100+8×1000)= 4×8×1×1+4×1×1×10+4×6×1×100+ 4×8×1×1000+9×8×10×1+9×1×10×10+ 9×6×10×100+9×8×10×1000+3×8×100×1+ 3×1×100×10+3×6×100×100+ 3×8×100×1000+1×8×1000×1+ 1×1×1000×10+1×6×1000×100+ 1×8×1000×1000= 32+040+2400+32000+720+0900+54000+ 720000+2400+03000+180000+2400000+ 08000+010000+0600000+08000000= **12013492**
Output (GPT-3.5):	The multiplication of 1394 by 8618 is equal to **12**,**013**,**6** **92**.
Output (GPT-4):	The multiplication of 1394 and 8618 equals **12**,**01** **4**,**05** **2**.
Correct Answer:	**12013492**

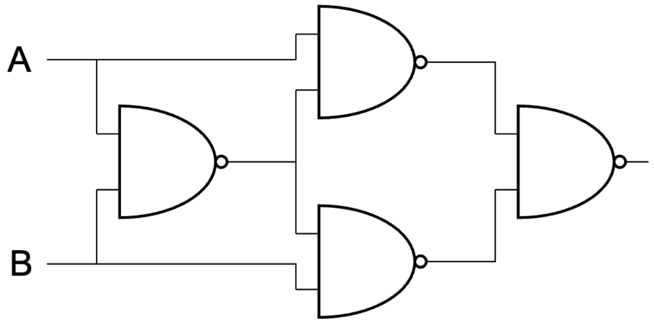
Length complexity T = 307

Model	Accuracy (exact match)	Accuracy (per-digit)
MLP-775M	96.9%	99.5 %
GPT-3.5	1.2%	61.85%
GPT-4*	5.3%	61.8%
Goat-7B*	96.9%	99.2 %

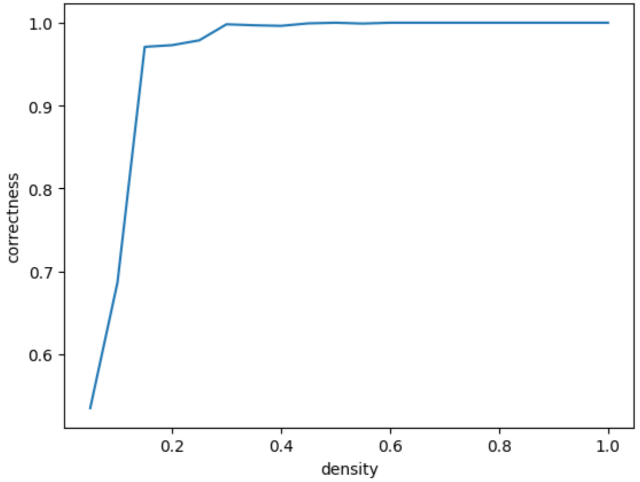
Experiment

$$f(\mathbf{x}) = \bigotimes_{i=1}^{n} \mathbf{x}_i$$

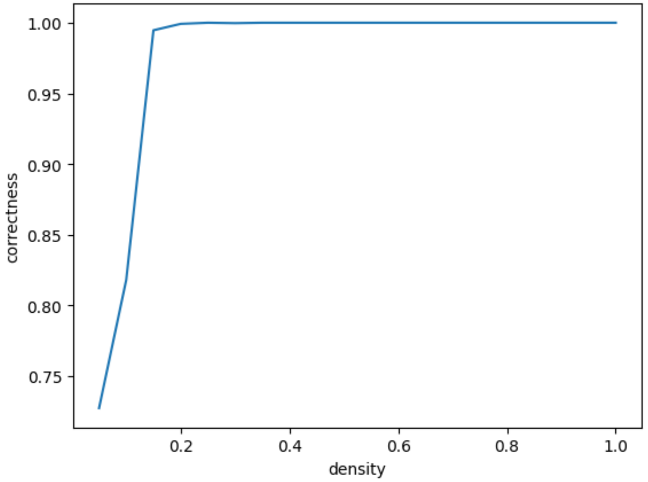
Multiple XOR function



XOR circuit with NAND gates



n=10, 20 densities, 50 cases / density



n=12, 20 densities, 50 cases / density

EPFL

Conclusions

- Every Turing computable function can be learned if long enough chain of thought is given.
- The main problem is the generation of the chain of thought data.
- Small amount of input data the model acts well => faster way to compute the samples.
- Chain of thought data from the internet makes this process faster than usal ML methods.

Thanks for your attention