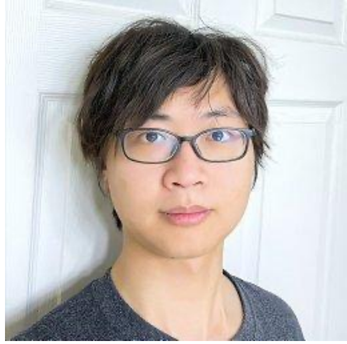
On the Turing Completeness of Prompting


Ruizhong Qiu
UIUC
(Presenter)

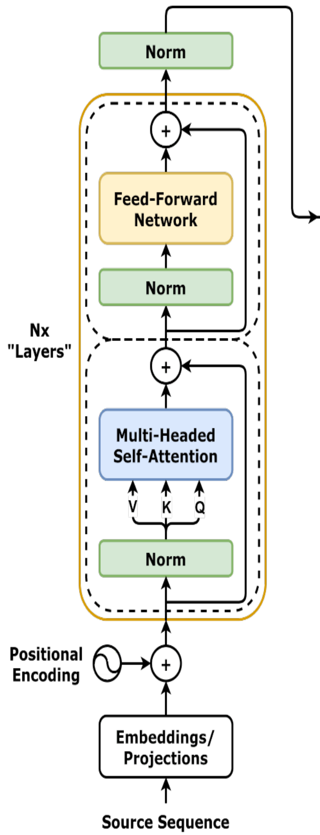

Zhe Xu
UIUC


Wenxuan Bao
UIUC


Hanghang Tong
UIUC

Expressive Power of Transformers

- Expressive power: what functions a model class can represent.
  - The mainstream architecture of LLMs is decoder-only **Transformers**.
- Existing studies: the classical **one-model-one-task** paradigm.
  - [PBM21]: The class of all hard-attention Transformers is Turing-complete.
  - [MS24]: can compute $\mathrm{TIME}\big(t(n)\big)$ functions in $\mathrm{O}\big(t(n)\big)$ chain-of-thought (CoT) steps.
- Practice: LLM prompting (i.e., the **one-model-many-tasks** paradigm).
  - Fundamentally, how powerful is the LLM prompting paradigm?



[PBM21] Jorge Pérez, Pablo Barceló, & Javier Marinkovic. Attention is Turing-complete. *Journal of Machine Learning Research 22*(75): 1–35, 2021.
[MS24] William Merrill & Ashish Sabharwal. The expressive power of Transformers with chain of thought. *International Conference on Learning Representations*, 2024.

ICLR

Main Results

- In this work, we show that prompting is in fact **Turing-complete**.
  - There exists a **single finite-size** Transformer Γ on which prompting is Turing-complete.
    - Not only existence: We give a **simple** and **explicit** construction.
  - **CoT complexity**: can compute $\mathrm{TIME}\big(t(n)\big)$ functions in $\mathrm{O}(t(n)\log t(n))$ CoT steps.
    - The single Transformer Γ is **nearly as efficient** as of the **class of all Transformers**, whose CoT complexity is $\mathrm{O}\big(t(n)\big)$.
  - **Precision complexity**: can compute $\mathrm{TIME}\big(t(n)\big)$ functions in $\mathrm{O}\Big(\log\big(n+t(n)\big)\Big)$ bits of precision.
    - The single Transformer Γ has the **same** precision complexity as that of the **class of all Transformers**.

Turing Completeness of Prompting (Theorem 3.1)

- Notation:
  - Let $\mathrm{generate}_{\Gamma}: \Sigma^+ \to \Sigma^+$ denote *autoregressive* generation with a Transformer $\Gamma: \Sigma^+ \to \Sigma$.
- There exist:
  - a finite alphabet Σ, a finite-size decoder-only Transformer $\Gamma: \Sigma^+ \to \Sigma$, and
  - coding schemes $\mathrm{tokenize}: \{0,1\}^* \to \Sigma^*$ and $\mathrm{readout}: \Sigma^* \to \{0,1\}^*$
- with which prompting is **Turing-complete**, in the sense that:
  - for every computable function $\varphi: \mathrm{dom}\,\varphi \to \{0,1\}^*$ with $\mathrm{dom}\,\varphi \subseteq \{0,1\}^*$,
  - there exists a prompt $\boldsymbol{\pi}_{\varphi} \in \Sigma^+$ such that for every input $\boldsymbol{x} \in \mathrm{dom}\,\varphi$,
  - $\mathrm{generate}_{\Gamma}\left(\boldsymbol{\pi}_{\varphi} \cdot \mathrm{tokenize}(\boldsymbol{x})\right)$ computes a finite CoT, and

$$\mathrm{readout}\left(\mathrm{generate}_{\Gamma}\left(\boldsymbol{\pi}_{\varphi} \cdot \mathrm{tokenize}(\boldsymbol{x})\right)\right) = \varphi(\boldsymbol{x}).$$

- Remarks:
  - Σ, Γ, tokenize, and readout are independent of the function φ;
  - The prompt $\boldsymbol{\pi}_{\varphi}$ is independent of the input $\boldsymbol{x}$;
  - For any $\boldsymbol{x} \in \{0,1\}^*$, tokenize & readout run in time $O(|\boldsymbol{x}|)$ & $O(|\varphi(\boldsymbol{x})|)$ on a RAM, respectively.

Proof Sketch

- How should we theoretically formulate **what prompting is**?
  - Natural languages are too unstructured.
  - Turing machines / common programming languages are a bit too complex.
  - Solution: a simple model of computation that is **nearly as efficient** as Turing machines.
- Construct a new model of computation: *2-tape Post–Turing machines* (2-PTMs).
  - 2-PTMs can be easily encoded into prompts using a **finite** alphabet.
  - Any $\mathrm{TIME}\big(t(n)\big)$ function can be computed by a 2-PTM in $\mathrm{O}(t(n)\log t(n))$ steps.
- Construct a **finite-size** Transformer Γ to simulate 2-PTMs via CoT steps.
  - Define CoT steps to record the execution of 2-PTMs.
  - The constructed Γ can compute any $\mathrm{TIME}\big(t(n)\big)$ function within:
    - $\mathrm{O}(t(n)\log t(n))$ CoT steps and
    - $\mathrm{O}\big(\log\big(n + t(n)\big)\big)$ bits of precision.

A Demonstrative Example

- **Construction**: A finite-size Transformer Γ over a finite-size alphabet Σ, where

  Σ = {#, AL, BL, AR, BR, A0, B0, A1, B1, A!, B!, A?, B?, -, +, @, ^, $, /, =, :, 0, 1}

- **Example**: Suppose φ decides the Dyck language (balanced parenthesis sequences) [Sch63].
  - The corresponding prompt π_φ for deciding the Dyck language:

    ```
    ^A?++++++++++++++@A0ALA0ALA?----@ARARA1ARBLB?++@A1#ARA?++++@B1BRB!+++@BL
    B!++++@B0ARB!-----------------------@ALARARA?--@A0ALA0ALA?----@ARARA1#$
    ```

  - The input 00 has **Shannon's** encoding [Sha56] S(00) = 1010, and it is tokenized as:

    tokenize(00) = ARARARARALALA1ALALA1=----------@

  - The generated CoT steps for computing φ(00):

    ```
    =++++++++++++++@AR/B1BR=+++@B0AR=-----------------------@=++++++++++++++@
    AR/B1BR=+++@B0AR=-----------------------@/A0ALA0AL=----@A0ALA0AL=----@A0
    ALA0AL/ARARA1ARBL=++@:0$
    ```

    - The final readout answer is :0$, which correctly computes φ(00) = 0 (because 00 ∉ Dyck).

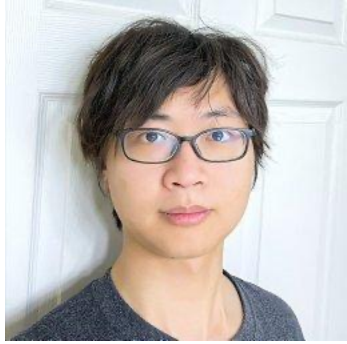
- More examples at:

  https://github.com/q-rz/ICLR25-prompting-theory/blob/main/main.ipynb

[Sch63] Marcel Paul Schützenberger. On context-free languages and push-down automata. *Information and Control 6*(3): 246–264, 1963.
[Sha56] Claude E. Shannon. A universal Turing machine with two internal states. *Automata Studies 34*: 157–165, 1956.

Thanks for watching

On the Turing Completeness of Prompting



Ruizhong Qiu
UIUC
(Presenter)

Zhe Xu
UIUC

Wenxuan Bao
UIUC

Hanghang Tong
UIUC