Simulating Weighted automata over Sequences and Trees

Michael Rizvi [1] Maude Lizaire [1] Clara Lacroce [2] Guillaume Rabusseau [1]

[1]Université de Montréal [2]McGill University

Introduction

- Liu et al. [1] showed transformers can **simulate DFA** with $\mathcal{O}\log T$ layers (even $\mathcal{O}(1)$ in some cases!)
- This result sheds light on **the algorithmic capabilities** of the transformer architecture

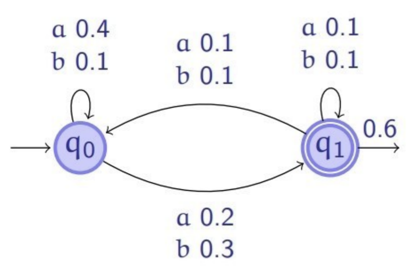
Weighted Finite Automata

A *weighted finite automaton* (WFA) of n states over Σ is a tuple $\mathcal{A} = \langle \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\beta}\rangle$, where

- $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^n$ are the initial and final weight vectors
- $\mathbf{A}^\sigma \in \mathbb{R}^{n\times n}$ is the matrix containing the transition weights associated with each symbol $\sigma \in \Sigma$

Every WFA $\mathcal{A}$ with real weights realizes a function $f_\mathcal{A} : \Sigma^* \to \mathbb{R}$, *i.e.* given a string $x = x_1 \cdots x_t \in \Sigma^*$, it returns $f_\mathcal{A}(x) = \boldsymbol{\alpha}^\top \mathbf{A}^{x_1} \cdots \mathbf{A}^{x_t}\boldsymbol{\beta} = \boldsymbol{\alpha}^\top \mathbf{A}^{x}\boldsymbol{\beta}$.

Example Consider the following WFA with **2** states on $\Sigma = \{a, b\}$



Operator Representation

$$\boldsymbol{\alpha} = \begin{bmatrix}1.0\\0.0\end{bmatrix} \quad \mathbf{A}^a = \begin{bmatrix}0.4 & 0.2\\0.1 & 0.1\end{bmatrix}$$

$$\boldsymbol{\omega} = \begin{bmatrix}0.0\\0.6\end{bmatrix} \quad \mathbf{A}^b = \begin{bmatrix}0.1 & 0.3\\0.1 & 0.1\end{bmatrix}$$

$$\mathrm{f}(\mathrm{ab}) = 0.4 \times 0.3 \times 0.6 + 0.2 \times 0.1 \times 0.6 = 0.084$$
$$= \boldsymbol{\alpha}^\top \mathbf{A}^a \mathbf{A}^b\, \boldsymbol{\omega}$$

Weighted Tree Automata

Binary Trees

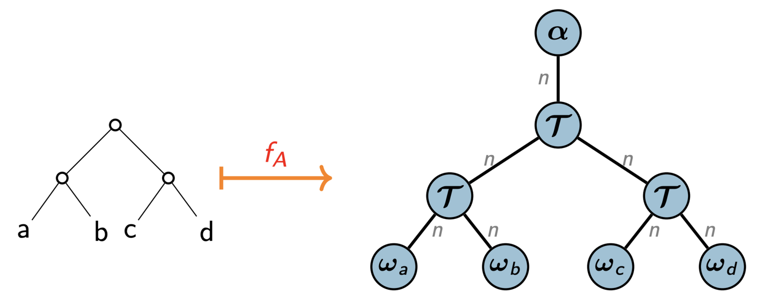
Given a finite alphabet Σ, the set of binary trees with leafs labeled by symbols in Σ is denoted by $\mathscr{T}_\Sigma$. Formally, $\mathscr{T}_\Sigma$ is the smallest set such that $\Sigma \subset \mathscr{T}_\Sigma$ and $(t_1, t_2) \in \mathscr{T}_\Sigma$ for all $t_1, t_2 \in \mathscr{T}_\Sigma$.

WTAs

A weighted tree automaton (WTA) $\mathcal{A}$ with n states on $\mathscr{T}_\Sigma$ is a tuple $\langle \boldsymbol{\alpha} \in \mathbb{R}^n, \boldsymbol{\mathcal{T}} \in \mathbb{R}^{n\times n\times n}, \{\mathbf{v}_\sigma \in \mathbb{R}^n\}_{\sigma\in\Sigma}\rangle$. A WTA $\mathcal{A}$ computes a function $f_\mathcal{A} : \mathscr{T}_\Sigma \to \mathbb{R}$ defined by $f_\mathcal{A}(t) = \langle \boldsymbol{\alpha}, \mu(t)\rangle$ where the mapping $\mu : \mathscr{T}_\Sigma \to \mathbb{R}^n$ is recursively defined by

- $\mu(\sigma) = \mathbf{v}_\sigma$ for all $\sigma \in \Sigma$,
- $\mu((t_1, t_2)) = \boldsymbol{\mathcal{T}} \times_2 \mu(t_1) \times_3 \mu(t_2)$ for all $t_1, t_2 \in \mathscr{T}_\Sigma$.

Example



The Transformer Architecture

The transformer architecture in our construction is similar to the **encoder in the original transformer architecture** [2]. The model is defined as follows

- Input: $X \in \mathbb{R}^{T\times d}$ where T is sequence length and d is embedding dimension
- Self-attention block:

$$f(\mathbf{X}) = \mathrm{softmax}(\mathbf{X}\mathbf{W}_Q\mathbf{W}_K^\top\mathbf{X}^\top)\mathbf{X}\mathbf{W}_V,$$

- Attention layer f_{attn}: h copies of f, concatenate the outputs
- Feedforward layer f_{mlp}: Simple feedforward MLP

Full L-layer model, with $f_{\text{tf}} : \mathbb{R}^{T\times d} \to \mathbb{R}^{T\times d}$:

$$f_{\text{tf}} = f_{\text{mlp}}^{(L)} \circ f_{\text{attn}}^{(L)} \circ f_{\text{mlp}}^{(L-1)} \circ f_{\text{attn}}^{(L-1)} \circ \ldots \circ f_{\text{mlp}}^{(1)} \circ f_{\text{attn}}^{(1)}.$$

Simulating WFA

Exact Simulation

Given a WFA $\mathcal{A}$ over some alphabet Σ, a function $f : \Sigma^T \to \mathbb{R}^{T\times n}$ *exactly* simulates $\mathcal{A}$ at length T if, for all $x \in \Sigma^T$ as input, we have $f(x) = \mathcal{A}(x)$, where $\mathcal{A}(x) = (\boldsymbol{\alpha}^\top, \boldsymbol{\alpha}^\top\mathbf{A}^{x_1}, \ldots, \boldsymbol{\alpha}^\top\mathbf{A}^{x_{1:T}})^\top$.

Approximate Simulation

Given a WFA $\mathcal{A}$ over some alphabet Σ, a function $f : \Sigma^T \to \mathbb{R}^{T\times n}$ *approximately* simulates $\mathcal{A}$ at length T with precision $\epsilon > 0$ if for all $x \in \Sigma^T$, we have $\|f(x) - \mathcal{A}(x)\|_F < \epsilon$.

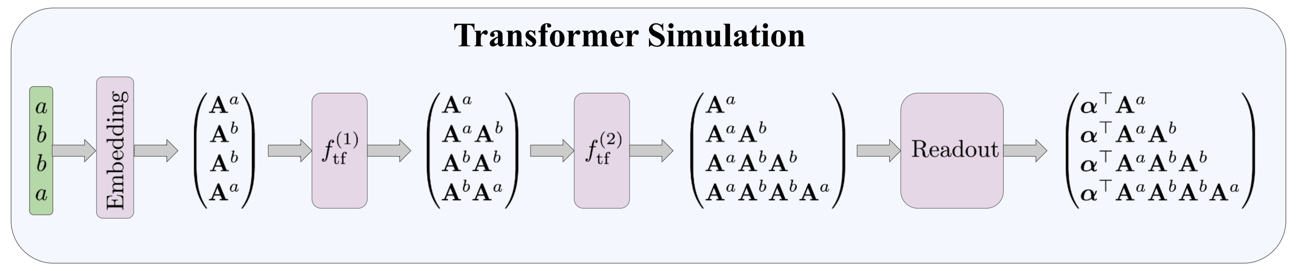


Figure 1. Simulation of the WFA computation over the input $w = abba$ with a transformer

Simulating WTA

Simulation by a function

Given a WTA $\mathcal{A} = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\mathbf{v}_\sigma\}_{\sigma\in\Sigma}\rangle$ with n states on $\mathscr{T}_\Sigma$, we say that a function $f : (\Sigma \cup \{[\![,]\!]\})^T \to (\mathbb{R}^n)^T$ simulates $\mathcal{A}$ at length T if for all trees $t \in \mathscr{T}_\Sigma$ such that $|\mathrm{str}(t)| \le T$, $f(\mathrm{str}(t))_i = \mu(\tau_i)$ for all $i \in \mathcal{I}_t$.

Simulation by a family of functions

We say that a family of functions $\mathcal{F}$ simulates WTAs with n states at length T if for any WTA $\mathcal{A}$ with n states there exists a function $f \in \mathcal{F}$ that simulates $\mathcal{A}$ at length T.

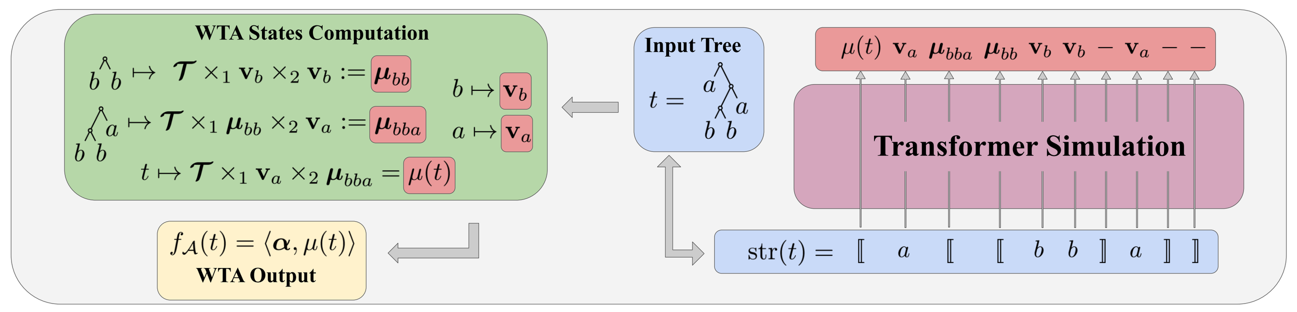


Figure 2. Computation of a WTA on the input tree $t = (a, ((b, b), b))$ (left) and simulation of the WTA computation over t with a transformer (right)

Main Theoretical Results

WFA

Theorem 1 Transformers using bilinear layers in place of an MLP and hard attention can *exactly* simulate all WFAs with n states at length T, with depth $\mathcal{O}(\log T)$, embedding dimension $\mathcal{O}(n^2)$, attention width $\mathcal{O}(n^2)$, MLP width $\mathcal{O}(n^2)$ and $\mathcal{O}(1)$ attention heads.

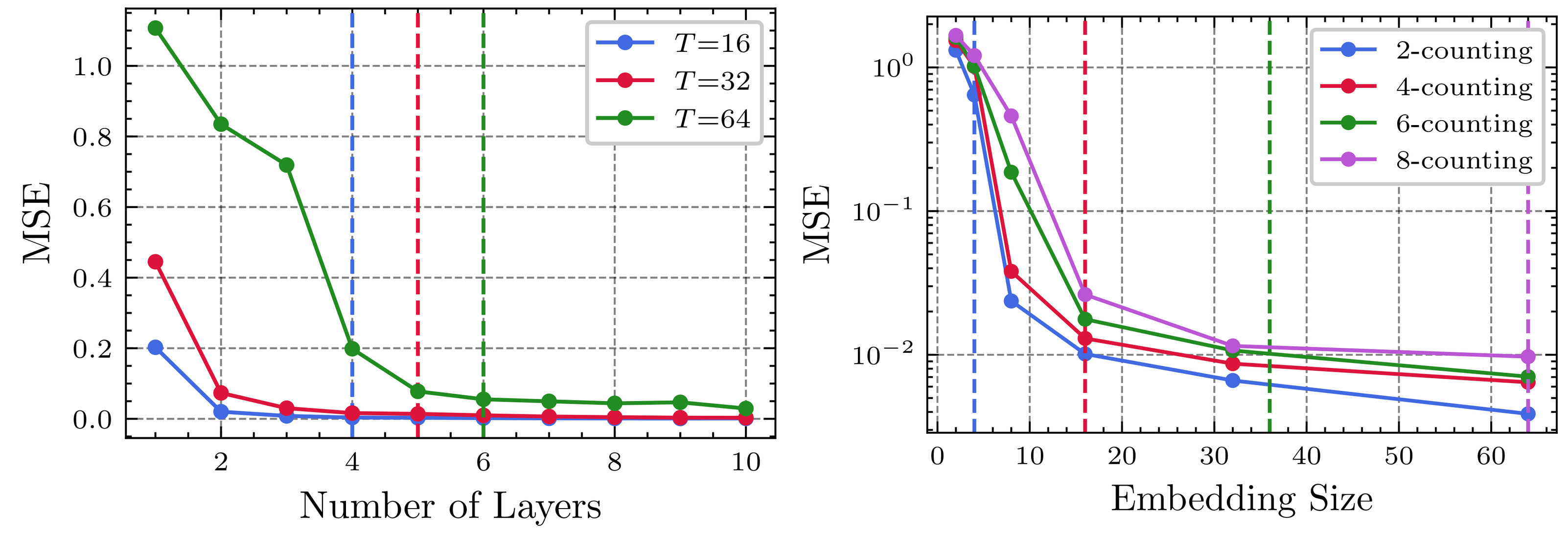
Theorem 2 Transformers can *approximately* simulate all WFAs with n states at length T, up to arbitrary precision $\epsilon > 0$, with depth $\mathcal{O}(\log T)$, embedding dimension $\mathcal{O}(n^2)$, attention width $\mathcal{O}(n^2)$, MLP width $\mathcal{O}(n^4)$ and $\mathcal{O}(1)$ attention heads.

WTA

Theorem 3 Transformers can *approximately* simulate all WTAs $\mathcal{A}$ with n states at length T, up to arbitrary precision $\epsilon > 0$, with embedding dimension $\mathcal{O}(n)$, attention width $\mathcal{O}(n)$, MLP width $\mathcal{O}(n^3)$ and $\mathcal{O}(1)$ attention heads. Moreover:

- Simulation over arbitrary trees can be done with depth $\mathcal{O}(T)$
- Simulation over balanced trees (trees whose depth is of order $\log(T)$) with depth $\mathcal{O}(\log(T))$.

Experimental Results



(a) Average MSE vs. number of layers (b) Average MSE vs. embedding size

Figure 3. Experimental results for simulation of counting automata. Right: we use an automaton which counts the number of 0s in $\Sigma = \{0, 1\}$ and vary the sequence length. Left: we use k-counting automata and vary the embedding size

Pautomac nb	4	12	14	20	30	31	33	38	39	45
num states	12	12	15	11	9	12	13	14	6	14
alphabet size	4	13	12	18	10	5	15	10	14	19
type	PFA	PFA	HMM	HMM	PFA	PFA	HMM	HMM	PFA	HMM
symbol sparsity	0.4375	0.3526	0.4944	0.3939	0.6555	0.3833	0.5949	0.7857	0.4167	0.8008
nb layers for ϵ	8	6	2	6	-	8	-	2	-	-

Table 1: Minimum number of layers to reach error $< \epsilon = 10^{-3}$

Discussion

- For both figures, increasing layers/embedding dimension **lowers the MSE**
- For Figure (a) this trend is **consistent with theory** (shown by the dotted lines)
- For Figure (b) stabilization **does not agree as closely** with our theoretical results (shown as dotted lines).

Conclusion

- We define simulation of **weighted automata** for **sequences and trees**
- We derive the notion of **approximate simulation** and how it applies to transformers
- We show that transformers can **simulate WFAs with $\mathcal{O}(\log T)$ layers**
- We show transformers can **simulate WTAs with $\mathcal{O}(\log T)$ layers**
- Our results extend the ones of Liu et al. for DFAs in **two directions**: from **boolean to real weights** and from **sequences to trees**

Future Work

- Our results mostly concern **expressivity** not **learnability**. Possibility to analyze learnability with **training dynamics analysis**
- Our results mostly provide **upper bounds**. It could be interesting to derive **lower bounds** on the expressivity of transformers

References

[1] Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All You Need. *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Scan me for paper link!