From Transformers to Weighted Automata: Towards the Verification of Large Language Models

Smayan Agarwal, Aslah Ahmad Faizi, Shobhit Singh, Aalok Thakkar
Ashoka University, India

DataMod 2025

Motivation

- Transformers are starting to be deployed in safety-critical applications (medical, legal, etc.)
- Their black-box nature hinders reliability and certification.
- Empirical benchmarks systematically inflate performance estimates
- Performance collapse on simple inductive reasoning tasks (InductionBench, Mirage).
- Reliance on superficial memorization rather than true understanding
- Counterintuitive scaling limits and arbitrary accuracy collapses on complex problems
- We need **formal, interpretable frameworks** for reasoning about model behaviour.

The Verification Challenge

- Verification using token sequences involves checking all inputs in the domain D. Even for a truncated domain $D = \Sigma^{\leq n}$, verifying a property for all words in D is computationally infeasible.
- Global constraints (e.g. Lipschitz bounds) involve a reduction in accuracy without providing verification certificates strong enough to justify the capability loss.

Our Contributions

1. **Transformer to Weighted Automata:** a formal reduction providing automata-theoretic foundation for LLMs.
2. **Identity Testing Algorithm:** tests whether a black-box string sampler is close to a WFA.

Why Automata-Theoretic Foundations

- Restricted classes of Transformers correspond to logical fragments (e.g. FO, FO[MOD], MSO), situating them within classical expressiveness hierarchies.
- Modeling soft attention requires probabilistic formalisms; **Weighted Finite Automata (WFA)** provide the natural extension.
- WFAs subsume deterministic automata and Hidden Markov Models, offering a uniform algebraic view of sequence models.
- The automata-theoretic setting comes with mature algorithmic tools: equivalence checking, minimization, spectral learning, and distance metrics between automata.

Step 1: Transformer to Structured RNN

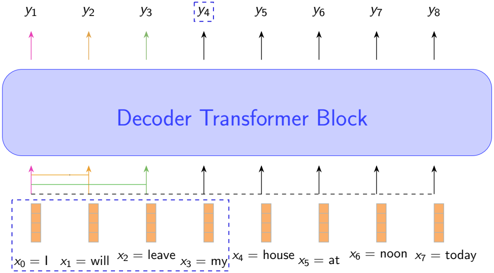
Transformer Structure: For the purpose of this paper, we use a **decoder-only** transformer with **one layer**, and a softmax activation function.

Step 1: Transformer to Structured RNN

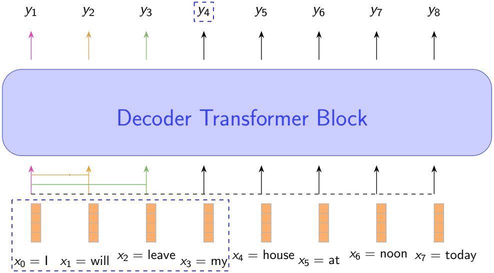
Transformer Structure: For the purpose of this paper, we use a **decoder-only** transformer with **one layer**, and a softmax activation function.
Decoder-only transformers are used for **next token prediction**, which are similar to the recurrent nature of RNN predictions.
At each token i, we will define an RNN whose ith block will have the same formulation as the transformer.

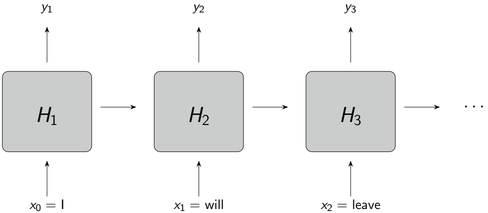
$$f_n = \frac{\sum_{j=1}^{n} v_j e^{q_n^\top k_j}}{\sum_{j=1}^{n} e^{q_n^\top k_j}}. \tag{1}$$

Step 1: Decoder vs. RNN



Step 1: Decoder vs. RNN





Step 1: Transformer to Structured RNN

Goal: Express softmax self-attention as a recurrent computation.

Construction

- Introduced by Zhang et al. 2024 [4]
- Maintain two accumulators per step: denominator and numerator.
- State $h_i = (g_i, f_i)$.
- Update via sigmoid-weighted recursion.
- Output $\beta(h_n) = f_n / e^{g_n}$ reproduces attention output.

Step 2: Structured RNN to Weighted Automaton

Algorithm 1: $\textsc{Convert-RNN-to-WA}(R,\ \theta,\ r)$

Input: RNN R, truncation length θ, target rank r.
Output: WA $A = (\nu, \{M_\sigma\}, \tau)$ of dimension r.

Set $P \leftarrow \Sigma^{\leq\theta}$, $S \leftarrow \Sigma^{\leq\theta}$ and index them arbitrarily.
for $p \in P, s \in S$ **do**
    $H[p, s] \leftarrow f_R(ps)$; `// Construct Hankel matrix`
end
Compute rank-r truncated SVD: $H \approx U_r \Sigma_r V_r^\top$
Define vectors $\nu \leftarrow U_r \Sigma_r^{1/2} e_\epsilon$ and $\tau \leftarrow (\Sigma_r^{1/2} V_r^\top)^\top e_\epsilon$; `// Initial/final vectors`
for $\sigma \in \Sigma$ **do**
    for $p \in P, s \in S$ **do**
        $H_\sigma[p, s] \leftarrow f_R(p\sigma s)$; `// Construct shifted matrix`
    end
    $M_\sigma \leftarrow \Sigma_r^{-1/2} U_r^\top H_\sigma V_r \Sigma_r^{-1/2}$; `// Project to get transitions`
end
return $A = (\nu, \{M_\sigma\}, \tau)$

From RNNs to Weighted Automata

- **Goal:** Distill a recurrent neural network (RNN) into a **Weighted Automaton (WA)** that approximates its sequence behavior.
- Both RNNs and WAs define functions $f : \Sigma^* \to \mathbb{R}$ mapping strings to scores or probabilities.
- WAs are **interpretable** and **linear**, while RNNs are black-box nonlinear systems.
- The standard approach here is **Spectral learning via Hankel matrices** [[1], [2]]

Fliess (1974)

Let S be a field. The rank of the Hankel matrix H_f associated with a function $f : \Sigma^* \rightarrow S$ is finite **iff** f is rational. In that case, there exists a Weighted Finite Automaton (WFA) A representing f with

$$|Q_A| = \text{rank}(H_f),$$

and no smaller WFA can represent f. [3]

Example: The Hankel Matrix

- Consider an alphabet $\Sigma = \{a, b\}$ and a function $f : \Sigma^* \to \mathbb{R}$ (e.g. the RNN's probability of each string).
- Select small prefix and suffix sets:

$$P = S = \{\varepsilon, a, b\}.$$

- The corresponding **finite Hankel block** is:

$$H_B = \begin{bmatrix} f(\varepsilon) & f(a) & f(b) \\ f(a) & f(aa) & f(ab) \\ f(b) & f(ba) & f(bb) \end{bmatrix}$$

What It Represents

Each entry $H_B(u, v)$ is the value of the function on the concatenated string uv. Rows correspond to prefixes, columns to suffixes and together they describe how the RNN continues a given prefix.

What is a Hankel Matrix?

- Formally, for any function $f : \Sigma^* \to \mathbb{R}$:

$$H_f(u, v) = f(uv), \quad u, v \in \Sigma^*.$$

- The full H_f is infinite; we work with a finite sub-block H_B defined by chosen prefixes P and suffixes S.
- If f is **rational**, the rank of H_f is finite, and this rank equals the number of states of the minimal Weighted Finite Automaton (WFA) representing f.
- Intuitively, each row of H_f corresponds to the predictive state reached after a prefix.

Building the Hankel Matrix from an RNN

- Treat the trained RNN as a black-box function $f(x)$ returning a score or probability for any string x.
- Choose finite prefix and suffix sets:

$$P = \text{prefixes}, \quad S = \text{suffixes}.$$

- For each pair (u, v) with $u \in P$, $v \in S$,

$$H_B(u, v) = f(uv).$$

  Query the RNN to fill all such entries.
- For each symbol $\sigma \in \Sigma$, build shifted blocks

$$H_\sigma(u, v) = f(u\sigma v),$$

  which capture how appending σ changes the predictive state.

From Hankel to Automaton via SVD

- Perform the **Singular Value Decomposition (SVD)** of the Hankel block:
$$H_B \approx U \Sigma V^\top.$$
- The top k singular directions define a k-dimensional subspace
- The WFA parameters are obtained from this factorization:
$$\alpha_0 = h_{\lambda,S} V \Sigma^{-1}, \quad \alpha_\infty = \Sigma^{-1} U^\top h_{P,\lambda},$$
$$A_\sigma = \Sigma^{-1} U^\top H_\sigma V \Sigma^{-1}.$$
- These give a linear system $A = (\alpha_0, \{A_\sigma\}, \alpha_\infty)$ such that
$$A(x) \approx f(x).$$

Finite Rank and Approximation in Practice

- In theory, the RNN defines an infinite Hankel matrix since one can query arbitrarily long strings.
- But since each row of H_B encodes the predictive distribution after a prefix u and longer prefixes (e.g. a, ab, abb, $abbb$, $\ldots$) yield increasingly similar conditional distributions the rows of H_B thus become nearly linearly dependent.
- Consequently, the RNN's behavior can be well-approximated by a finite-rank WFA.

Approximation Guarantees

- **Theoretical:**

  If the empirical Hankel matrix $\widehat{H}$ satisfies

  $$\|H - \widehat{H}\|_2 \leq \eta,$$

  then the reconstructed automaton A obeys

  $$|f_A(w) - f_R(w)| \leq C(\theta, |\Sigma|, r)\, \eta, \qquad \forall\, w \in \Sigma^{\leq \theta}.$$

- Small perturbations in H lead to controlled changes in its SVD factors (U, Σ, V) (Davis–Kahan, 1970), hence bounded error in the reconstructed WA.
- *If the Hankel approximation is accurate to within η, the automaton is accurate to within $O(\eta)$.*

  θ: truncation length $\qquad$ r: target rank (number of WFA states)

Approximation Guarantees

Empirical: (Eyraud & Ayache, 2020) Spectral distillation from RNNs yields high-fidelity WFAs:

- NDCG > 0.9 : similarity of next-symbol ranking between RNN and WA,
- WER-D < 0.2 : word error rate measuring sequence-level reconstruction accuracy.

Complexity Analysis

- Let $k = |\Sigma^{\leq\theta}|$i.e. number of strings smaller than cutoff length
  - $O(k^2)$ evaluations of f_R, each $O(\theta \cdot T_{\text{step}})$ time,
  - $O(k^3)$ time for SVD of a $k \times k$ matrix,
  - $O(|\Sigma| \cdot k^2)$ memory for Hankel and shifted blocks.
- Exponential growth in θ limits scalability; sampling or low-rank methods mitigate the cost.

The Identity Testing Problem

Goal

Given sample access to a black-box distribution P and known rational stochastic language Q, test if $\|P - Q\|_1 < \varepsilon$.

- Provides statistical certificate of similarity to a WFA.
- Enables verification even for proprietary black-box models.

Key Idea: Truncation to Finite Support

- We already have a rich theory of distance estimation between known and unknown distributions on **finite supports**.
- For infinite domains, we **truncate** the distribution so that the tail error is bounded by a fraction of ε.
- This reduces the infinite domain Σ^* to a finite set $\Sigma^{\leq\theta}$.
- Select $\theta(Q, \varepsilon/3)$ such that the tail mass is less than $\varepsilon/3$.
- Finally, apply a **finite-domain tolerant tester** (Canonne et al., 2022) on the truncated head.

Visualization

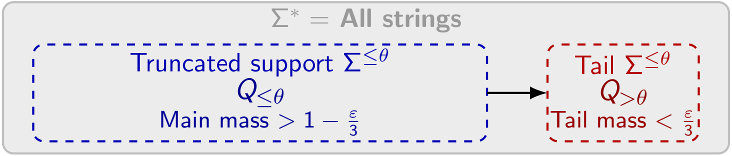


Figure: Schematic of truncation and tail error decomposition. The gray rectangle represents the full domain Σ^*. The blue box is the truncated support $\Sigma^{\leq\theta}$, containing most of the probability mass ($Q_{\leq\theta} > 1 - \varepsilon/3$). Truncation ensures the tail mass $Q_{>\theta}$ is bounded by $\varepsilon/3$, and hence contributes minimally. The total ℓ_1 error is decomposed as γ (estimation) $+C\eta$ (spectral) $+\delta$ (tail mass), highlighting that the bulk of the distribution is concentrated in a manageable region while the tail is small and controlled.

Algorithm 2: ℓ_1 Identity Tester

1. Compute $\theta(Q, \varepsilon/3)$.
2. Draw N i.i.d. samples from model P.
3. Reject if too many long strings ($|x| > \theta$).
4. Run tolerant identity tester on truncated supports.

Sample Complexity:

$$N = \Theta\left(\frac{\sqrt{k}}{\varepsilon^2} + k \log k \log \frac{1}{\delta}\right), \quad k = |\Sigma^{\leq \theta}|.$$

The Computational Barrier

- Hankel matrix size $O(|\Sigma|^{2\theta})$ limits scalability.
- Reflects trade-off: expressivity vs. tractability.

Future Directions

- Can the exponential dependence on the truncation length be avoided while preserving approximation quality?
  - Low rank approximations of Hankel matrices?
  - Randomised algorithms that avoid constructing the full Hankel matrix?
  - Leveraging patterns in transformer architecture?
- How do architectural choices affect the complexity of automata based approximation?
- What is the minimal class of weighted automata needed to approximate practical transformers?
- Implementation on real world transformers
- Creation of evaluation and verification metrics that replace empirical benchmarks

Conclusion

- formal reduction from decoder-only transformers to WFA.
- Identity testing algorithm for rational stochastic languages.
- Establishes automata-theoretic foundations for LLM verification.
- Opens pathway for rigorous, explainable model analysis.

Thank You!

Questions?

References I

Borja Balle and Mehryar Mohri.
Learning weighted automata.
In *International Conference on Algebraic Informatics*, 2015.

Rémi Eyraud, Colin de la Higuera, and Pascale Sébillot.
Distillation of weighted automata from recurrent neural networks using a spectral approach.
In *Machine Learning*, 2024.

Michel Fliess.
Matrices de hankel.
Journal de Mathématiques Pures et Appliquées, 53:197–222, 1974.

Yuhao Zhang, Aws Albarghouthi, and Loris D'Antoni.
A one-layer decoder-only transformer is a two-layer rnn: With an application to certified robustness, 2024.