# Report: AdaAX - Explaining RNNs with Adaptive Automata (Paper Structure)


## Abstract

Recurrent Neural Networks (RNNs) are effective for sequential data but lack transparency. This paper proposes **AdaAX**, a method to explain RNNs by constructing a Deterministic Finite Automaton (DFA). Unlike prior methods that fix state partitions early, AdaAX forms DFA states *adaptively*. It identifies fine-grained "core sets" based on RNN transition patterns and merges them strategically, allowing a trade-off between the explanation's fidelity (accuracy) and complexity (size). Experiments demonstrate AdaAX achieves higher fidelity with significantly smaller DFAs compared to baselines.


## 1. Introduction

- **Problem:** RNNs are powerful but function as "black boxes," making it hard to understand or trust their decision-making process. Interpretable models are needed.
- **Proposed Solution:** Use a DFA as an interpretable proxy model for an RNN. States in the DFA abstract RNN hidden states, and transitions follow input symbols. Paths (patterns) to accepting states explain predictions.
- **Limitations of Existing Work:** Current DFA extraction methods often pre-partition the RNN's hidden state space, leading to DFAs that are either inaccurate (low fidelity) or too large and complex to understand.
- **Contribution (AdaAX):**
  - A novel DFA extraction method using **adaptive states** formed by merging fine-grained core sets.
  - Decouples pattern identification (high fidelity) from state formation (controlled complexity).
  - Provides a mechanism to explicitly trade fidelity for lower complexity.
  - Achieves superior performance (higher fidelity, lower complexity) experimentally.

## 2. Preliminaries

- **Recurrent Neural Network (RNN):** Processes sequences $x = (x_1, ..., x_T)$, $x_t \in \Sigma$ (alphabet), computing hidden states $h_t = g(h_{t-1}, x_t)$ and a final output $y = f(h_T)$. $\mathbb{H}$ denotes the hidden state space.
- **Deterministic Finite Automaton (DFA):** A tuple $\mathcal{H} = (Q, \Sigma, \delta, q_0, F)$:
  - $Q$: Finite set of states.
  - $\Sigma$: Alphabet (same as RNN).
  - $\delta$: Transition function ($Q \times \Sigma \to Q$).
  - $q_0$: Start state (representing RNN's $h_0$).
  - $F \subseteq Q$: Set of accepting states.

- **RNN Explanation via DFA:** The DFA $\mathcal{H}$ explains the RNN $\mathcal{R}$ if its state transitions and acceptance behavior approximate the RNN's hidden state dynamics and final predictions.
- **Patterns:** Input sequences $p$ such that $\delta(q_0, p) \in F$. They represent inputs leading to the target prediction.
- **Problem Definition:** Given an RNN $\mathcal{R}$ and data $\mathcal{D}$, learn a DFA $\mathcal{H}$ that maximizes **fidelity** and minimizes **complexity** (size $|Q|$).
  - **Fidelity:** Measures prediction agreement between $\mathcal{H}$ and $\mathcal{R}$.

    $$fidelity(\mathcal{H}) = \frac{\sum_{x \in \mathcal{D}} \mathbb{I}(\mathcal{R}(x) = \mathcal{H}(x))}{|\mathcal{D}|}$$

    (Eq. 2)
  - **Accepting States (F):** Often correspond to RNN hidden states leading to a specific class prediction.

    $$F_{RNN} = \{h \in \mathbb{H} \mid f(h, x) = 1, \forall x \in \Sigma\}$$

    (Eq. 1)
    $F_{DFA}$ contains abstract states representing $F_{RNN}$.

## 3. Related Work

(This section summarizes the context inferred from the paper's motivation) * Existing methods for extracting DFAs from RNNs often rely on clustering RNN hidden states (e.g., using K-means) *before* learning transitions. * This pre-partitioning can be suboptimal, as clusters based solely on proximity might not align well with the actual transition dynamics learned by the RNN. * Such methods can result in low-fidelity DFAs or require a very large number of states (high complexity) to capture the RNN's behavior accurately.

## 4. The AdaAX Method

AdaAX employs a three-step process with adaptive state formation:

### 4.1 Step 1: Clustering (Initial Grouping)

- Collect hidden states from the RNN using training data $\mathcal{D}$.
- Perform an initial, coarse clustering (e.g., K-means) on the hidden states $\mathbb{H}$.
- Treat the start state ($h_0$) and accepting states ($F_{RNN}$) as distinct initial groups.
- *Purpose:* Primarily for efficiency in the pattern extraction step, not for defining final DFA states.

### 4.2 Step 2: Pattern Extraction (Backward Search & Core Sets)

- Performs a backward, depth-first search from the accepting states $F_{RNN}$ towards the start state $h_0$.

- Identifies **Core Sets:** For a focal set of states $C$ and an input symbol $x$, the core set consists of preceding states $h$ such that $g(h, x) \in C$.
  - Concept of preceding states P(C):

    $$P(C) = \{h \in \mathbb{H} \mid g(h, x) \in C, x \in \Sigma\}$$

    (Based on Eq. 3)
  - Core sets group states based on *shared transition behavior*, providing finer granularity than initial clusters.
- Traces paths (sequences of symbols) back to $h_0$, defining **patterns**.
- **Pruning:** Patterns with low support (frequency in $\mathcal{D}$) below a threshold $\theta$ can be removed.
  - Pattern Support:

    $$supp_{\mathcal{D}}(p) = \frac{\sum_{x \in \mathcal{D}} y(p, x)}{|\mathcal{D}|}$$

    (Definition 2.4)

### 4.3 Step 3: Consolidation (DFA Construction & Merging)

- Builds the DFA $\mathcal{H}$ iteratively by adding extracted patterns (typically sorted by support).
- Incorporates core sets and transitions from each pattern into the DFA.
- **Adaptive State Merging:** To control complexity, newly added core sets ($q_t$) are evaluated for merging with existing DFA states ($S \in Q_t$).
  - Find **Neighboring States** $\mathcal{N}(q_t, Q_t)$:

    $$\mathcal{N}(q_t, Q_t) = \{S \in Q_t \mid d(q_t.h, S.h) < \tau\}$$

    (Eq. 5)
    where $q_t.h$ is the RNN hidden state value corresponding to $q_t$ (related to Eq. 4: $q.h = f(f(f(h0, p_1), p_2) \ldots, p_l)$ ) and $\tau$ is a distance threshold.
  - Merge $q_t$ with the closest neighbor $S \in \mathcal{N}(q_t, Q_t)$ *only if* the estimated drop in fidelity is below a user-defined threshold $\Delta$.
  - Merging combines prefixes and handles outgoing transitions intelligently.
  - This merging process forms the final **adaptive states** of the DFA, balancing fidelity and complexity.

## 5. Experiments

- **Setup:** AdaAX compared against baseline DFA extraction methods. LSTMs trained on various datasets.
- **Datasets:** Included synthetic data (e.g., Tomita grammars, other regular languages) and real-world data (e.g., Yelp reviews, MIMIC-III health records, Educational Process Mining).

- **Results:**
  - **Fidelity vs. Complexity:** AdaAX consistently produced DFAs with higher fidelity for a given complexity (number of states) or significantly lower complexity for comparable fidelity, compared to baselines.
  - **Effectiveness of Merging:** The consolidation step effectively reduced DFA size while preserving high fidelity.
  - **Sensitivity Analysis:** AdaAX showed better sensitivity in identifying "flip points" – minimal input changes causing prediction flips.
  - **Case Studies:** Demonstrated utility in understanding model behavior on specific datasets (e.g., diagnosing RNN failures).

## 6. Conclusion

AdaAX presents a novel approach for extracting explanatory DFAs from RNNs. By introducing **adaptive states** formed through identifying fine-grained **core sets** and then consolidating them via a fidelity-controlled **merging** process, AdaAX overcomes limitations of prior methods. It generates more accurate (higher fidelity) and simpler (lower complexity) explanations, providing a valuable tool for interpreting RNN behavior.