Interpretable Predictive Models for Healthcare via Rational Multi-Layer Perceptrons

THITI SUTTAKET and STANLEY KOK, Department of Information Systems and Analytics, National University of Singapore, Singapore

The healthcare sector has recently experienced an unprecedented surge in digital data accumulation, especially in the form of electronic health records (EHRs). These records constitute a precious resource that Information Systems (IS) researchers could utilize for various clinical applications, such as morbidity prediction and risk stratification. Recently, deep learning has demonstrated state-of-the-art empirical results in terms of predictive performance on EHRs. However, the blackbox nature of deep learning models prevents both clinicians and patients from trusting the models, especially with regards to life-critical decision making. To mitigate this, attention mechanisms are normally employed to improve the transparency of deep learning models. However, these mechanisms can only highlight important inputs without sufficient clarity on how they correlate with each other and still confuse end-users. To address this drawback, we pioneer a novel model called Rational Multi-Layer Perceptrons (RMLP) that is constructed from weighted finite state automata. RMLP is able to provide better interpretability by coherently linking together relevant inputs at different timesteps into distinct sequences. RMLP can be shown to be a generalization of a multi-layer perceptron (that only works on static data) to sequential, dynamic data. With its theoretical roots in rational series, RMLP’s ability to process longitudinal time-series data and extract interpretable patterns sets it apart. Using real-world EHRs, we have substantiated the effectiveness of our RMLP model through empirical comparisons on six clinical tasks, all of which demonstrate its considerable efficacy.

CCS Concepts: • Applied computing → Health informatics.

ACM Reference Format:

1 Introduction

The widespread adoption of information systems in past decades, coupled with its associated digitalization of all kinds of media, has led to the accumulation of a large amount of data, a precious resource that can potentially serve as fodder for machine learning systems.

The healthcare domain has experienced a rapid growth in the amount of digital data collected in many forms such as data from wearable sensors [77], or patient electronic health records (EHRs). The adoption of EHRs in particular has improved the quality of medical care [41]. EHR systems in hospitals now routinely record all patient-related information such as demographic details, physiological measurements, diagnoses, medications, laboratory tests, surgical procedures, and visit dates [11]. These digitized data in healthcare present an opportunity for IS researchers to utilize their patient-specific data for various clinical applications such as readmission prediction [6, 10, 76].

Authors’ Contact Information: Thiti Suttaket, e0384140@u.nus.edu; Stanley Kok, skok@comp.nus.edu.sg, Department of Information Systems and Analytics, National University of Singapore, Singapore.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 ACM.
ACM 2158-6578/2024/0-ART0
https://doi.org/Xxxxxxxx.Xxxxxxxx

ACM Trans. Manag. Inform. Syst., Vol. 0, No. 0, Article 0. Publication date: 2024.
risk profiling [40], chronic disease management [69], modeling of activities of daily living in senior care [77], multi-disease predictive analytics [57] and cost reduction in clinical decision-making [26]. One of the most important classes of applications among these is health risk prediction in which we are interested in estimating the risk that an individual develops a specific disease (e.g., diabetes) or is afflicted by a clinical event (e.g., death). This allows medical practitioners to prioritize high risk individuals to receive precious medical resources [7, 20, 56]. For example, Clalit, the largest of Israel’s four state-mandated health service organizations, has adopted machine learning approaches to identify high risk individuals for clinical events such as flu complications, COVID-19, and deterioration in the elderly [20]. For the task of health risk prediction, deep learning [29], one of the most prominent machine learning models, has achieved state-of-the-art empirical results, as demonstrated by RETAIN [15] and Adacare [46].

Even though these deep learning models have decent predictive performances, their acceptance in clinical settings is still limited because of trust and transparency issues [1]. With the lack of transparency in the models, a medical practitioner cannot explain their results to patients and is therefore unable to guarantee the safety of a patient [47]. In this regard, the need for explanations (other than predictions) has been raised by the IS community [2].

To enhance the interpretability of deep learning models, post-hoc explanations can be employed. Such an approach (e.g., LIME [60]) is model-agnostic, i.e., it can be applied to models other than deep learning ones. It involves learning another interpretable model (usually a simple one like a decision tree) from the predictions of the blackbox model of interest so as to mimic the blackbox model. The ability of the interpretable model to represent the blackbox model is called fidelity or faithfulness. However, if the decision function of the blackbox model is overly complex, the simple model may be ill-suited to imitate the blackbox model well enough in terms of the predictions made (i.e., the simple model has low fidelity to the blackbox model).

Another approach, which does not require simplifying the black-box model, focuses on determining the relevance of individual input variables by assigning scores to each variable to indicate their contribution to the model’s predictions. An example of such a technique is Layer-wise Relevance Propagation or LRP [4], which is designed to explain individual predictions made by feed-forward neural networks in terms of their input variables. It operates by propagating the relevance from the output layer back to the input layer. This propagation approximates relevance through a sequence of locally performed Taylor expansions at each neuron [50]. Similarly, SHapley Additive exPlanations (SHAP) [44] assigns feature importance scores based on Shapley values from cooperative game theory, highlighting how each feature contributes to model output across various combinations.

In contrast to the post-hoc methods previously discussed, which aim to extract interpretation from trained models, certain models are purposefully designed with interpretability as a core consideration from the outset. One prevalent technique is attention mechanism [15, 39, 45, 46, 51] which weight inputs at each time point in order to highlight those that are most relevant to a prediction task. However, only highlighting which inputs are important without sufficient clarity on how they correlate with each other and with the prediction task causes end-users to be confused [53]. For example, in predicting whether an individual would develop lung cancer, an attention mechanism could simultaneously highlight the important features as the individual’s HIV status and smoking habit. This could mislead users into thinking that both of these are correlated when, in reality, research has shown them to be uncorrelated risk factors [65]. These existing models are not designed to capture sequential correlations due to the challenges in model designing. Each attention weight operates independently, resulting in a lack of information exchange among individual attention weights. To compound the problem, the attention weights are frequently so diffused that a multitude of inputs are concurrently highlighted across time points. Not only would this
obfuscate the identities of the truly important inputs, but would also result in information overload for the end-users.

Post-hoc methods like SHAP and LIME encounter this similar limitation in extracting sequential correlation. Not only are SHAP and LIME unsuitable for handling sequential data, but they also interpret the results from a trained model that may not have been designed to capture serial correlations among features. Therefore, it is not essential for them to contain this resulting representation. Moreover, both SHAP and LIME interpret results locally, lacking consideration for global correlations between features in the data. For instance, in LIME, for each instance of interest, it requires small perturbations to the selected instance, generating new varied instances. A user-defined surrogate model is then fitted to these instances to give explanations.

On the other hand, weighted finite state automaton (WFSA) do not suffer from the interpretation issue associated with correlation in sequence data because of its ability to capture sequential patterns. A WFSA is widely used for processing text data and its recent neural versions (such as neural WFSA [62]) can be utilized to capture sequential input patterns in text classification tasks. A pattern is a sequence of words and wildcards that can match against a specific text span by replacing wildcards with concrete words. Schwartz et al. (2018; [62]) further relaxed the definition of a pattern by allowing some words to be dropped, inserted, or replaced with similar words. In this manner, each pattern, learned from input text data, shows how different parts of the input text correlate with each other. For example, consider the input text below.

```
Robert studied in a school that taught in English but learned to speak Spanish well too.
```

Suppose two patterns are used for making a prediction of whether a subject (Robert) is multilingual. The first pattern is the bolded parts of the input (i.e., studied in English) and the second pattern is the underlined parts (i.e., speak Spanish well). We therefore know that the second, third, and ninth words (from the 1st pattern) are correlated to reflect his English ability. On the other hand, words 14 - 16 (from the 2nd pattern) are correlated to show his Spanish ability. This can be contrasted with an attention mechanism, which simultaneously highlights all the important locations of the input sequence (in this case, words 2, 3, 9, and 14-16) without showing how the individual locations are correlated as shown in the following input text.

```
Robert studied in a school that taught in English but learned to speak Spanish well too.
```

This example of textual data can serve as a useful parallel for clinical data, in which each word corresponds to a clinical event (or a set of clinical events) observed at a time point. For example, two consecutive words in the textual data are equivalent to clinical data collected at two consecutive time points.

To utilize the representational power of WFSA, neural WFSA [62] combines deep learning with WFSA. However, it is based on logistic regression and thus is limited to capturing linear interactions among inputs. In order to model complex combinations of features in a non-linear manner, we propose a model called Rational Multi-Layer Perceptrons (RMLP) that can be viewed as an extension of multi-layer perceptrons (MLPs) to sequential data. Like neural WFSAs, it provides interpretable patterns as WFSAs that link relevant inputs at different timesteps. This allows RMLP to support many clinical use cases in which it is essential to derive non-linear features for prediction tasks. For example, we would like to learn from data that the normal range of systolic blood pressure (90-120 mmHg) is predictive of good health. Note that a linear model cannot represent this range because it can only capture a monotonic relationship between its feature and output, and cannot represent the maximum or minimum of the range. (A MLP can easily represent this through multiple layers of ReLU.)
In this paper, we follow the design science paradigm of developing useful artifacts to address practical problems [32]. Given the pressing need of medical practitioners for an effective yet interpretable predictive model for EHR data, we pioneer the Rational Multi-Layer Perceptron (RMLP). Our RMLP model subsumes vanilla MLPs as its special case (and hence can model complex combination of features in a non-linear manner), and generalizes the static nature of MLPs so as to handle the temporal and sequential information present in longitudinal data. The mathematical proof is provided in Theorem A.10 of Appendix A.1.3.

In addition to presenting a mathematical proof illustrating the extension of MLP to sequential data, we establish a broader principle for extending static models to sequential data with WFSA, as detailed in Theorem A.8 of Appendix A.1.1. This overarching principle represents one of our primary contributions, providing a foundational framework for transitioning from a static model to a dynamic one using WFSA. Our contribution bridges a significant gap in the prior literature [62].

According to the design science research framework, our proposed model is the designed artifact, which improves the existing foundations or methodologies in the design science knowledge-base. We accomplish this by demonstrating how RMLP extends a multi-layer perceptron to handle sequential data and subsumes neural WFSAs as its special case. Additionally, we offer a rigorous proof that serves as a guideline for transforming a static model into one that processes sequential data. Moreover, we highlight the superior interpretability of RMLP compared to the attention mechanisms commonly utilized in deep learning approaches.

In sum, our contributions are as follows.

• We propose a novel, interpretable model called RMLP for predictive healthcare tasks.\(^1\)
• We rigorously prove how RMLP is a generalization of a multi-layer perceptron (MLP) to sequential data and how it encompasses neural WFSAs as its specific instance. Furthermore, we offer a broader proof that serves as a guiding principle for extending a static model to sequential data.
• We empirically compare our RMLP model to state-of-the-art baselines, and show that RMLP outperforms them in terms of predictive accuracy on six real-world medical tasks.
• We qualitatively showcase the interpretability of the patterns RMLP learns at both personalized individual and coarse-grained (sub)population levels, and elucidate the “reasoning” that each pattern represents.

2 Related Work

Being able to make accurate predictions for clinical events (such as mortality or disease onset) is an important problem because of its potential to improve healthcare outcomes and reduce operational cost. The traditional approaches to tackle this problem are largely based on disease severity classification systems. In such systems, a list of criteria indicative of a disease (or clinical event such as death) is identified, and a score is assigned based on the number of criteria that a patient satisfies. SAPS [12] is a classification system used to evaluate the mortality risk for post-cardiac arrest patients. APACHE [78] has been developed to quantify mortality risk in intensive-care units using clinical factors such as care procedures administered and medications prescribed. LACE [73] evaluates the risk of readmission and death after discharge from a hospital. HOSPITAL Score [22] is used to evaluate the risk of 30-day readmission.

More recently, sophisticated deep learning approaches have been applied to clinical prediction [13, 16, 31, 45]. Choi et al. (2016b; [16]) use recurrent neural networks (RNNs) to predict in-patient mortality upon hospital admission using historical medical records. Che et al. (2018; [13]) improve upon this by imputing missing data. Ma et al. (2017; [45]) use a more sophisticated deep learning

\(^1\)The source code is available at https://github.com/tsuttaket2/RMLP.
model based on bidirectional RNNs to make predictions. Since these deep learning models are sequential models, they can leverage the temporal, sequential nature of medical events [51]. In contrast, traditional static machine learning models can only aggregate temporal input data into coarse-grain features (e.g., mean and standard deviation). Thus they cannot completely leverage the sequential information in temporal medical events. To illustrate, consider two patients: one who has recently experienced a rise in blood pressure and another whose blood pressure increased and then returned to normal. Intuitively, the former patient, whose blood pressure has recently risen, is at a higher risk of experiencing clinical adversity. However, when aggregating historical blood pressure data from these individuals for traditional static machine learning models, the aggregated blood pressure measurements of both cases can end up being similar despite differing histories. This leads to the loss of valuable temporal information, which can be crucial for making accurate predictions in healthcare scenarios.

Attention mechanisms in deep learning [39, 45] have been used to measure the relationships among different clinical visits. Sun et al. (2018; [68]) highlight the relative importance of clinical features via an adversarial-attack approach in a deep learning framework. RETAIN [15] is a state-of-the-art deep learning system that uses a two-level neural attention mechanism upon an RNN to model longitudinal EHR data and capture temporal information. To generate attention weights for interpretability, RETAIN uses two RNNs. The first RNN (RNN $\alpha$) uses the softmax function to get visit-level attention weights, and the second RNN (RNN $\beta$) uses the tanh function to calculate clinical-variable-level attention weights. Adacare [46] is another state-of-the-art deep learning approach that also uses an attention mechanism for interpretability. It uses dilated convolutions with multi-scale receptive fields to capture the long and short-term historical variations in longitudinal EHR data.

Both RETAIN and Adacare take a step towards interpretability through their use of neural attention mechanisms. They are able to identify important features at each time step across a time span. However, they do not elucidate the connections among the features at different time steps, which are often observed in real-world healthcare scenarios. For instance, consider the treatment recommendations for type II diabetes by the American Diabetes Association, which follow a step-wise progression starting with lifestyle interventions and the drug metformin, followed by the addition of a sulfonylurea medication if metformin is insufficient for glucose control. Subsequently, basal insulin will be introduced, and in more intensive cases, intensive insulin therapy is required [52]. In scenarios where diabetes patients are susceptible to various complications, being able to distinguish such sequential treatment plans for diabetes from other medications becomes essential for result interpretability.

In contrast to this limitation, our RMLP model learns interpretable patterns that thread together features at different time steps into a coherent whole. For example, RETAIN’s and Adacare’s attention mechanism can identify important features $A_1$ and $B_1$ at time-step 1 and important features $A_2'$ and $B_2'$ at time step 2. However, they do not know how a feature at time-step 1 cohere with a feature at time-step 2. Does $A_2'$ follow $A_1$, i.e., $A_1 \rightarrow A_2'$? Or $A_1 \rightarrow B_2'$? Or do combinations of the features interact, e.g., $(A_1, B_1) \rightarrow B_2'$ and $B_1 \rightarrow (A_2', B_2')$? Or perhaps all of the above? Our RMLP model can learn patterns to coherently capture all such regularities.

3 Background

Our model RMLP is built upon semirings and weighted finite state automata. We describe each of them in turn.
3.1 Semirings

A semiring is defined as a set \( \mathbb{K} \) that is associated with two binary operations, \( \oplus \) (addition) and \( \otimes \) (multiplication), and two identity elements, \( 0 \) (for addition) and \( 1 \) (for multiplication). A semiring requires both \( \oplus \) and \( \otimes \) to be associative, \( \oplus \) to be commutative, \( \otimes \) to distribute over addition, and \( 0 \) to annihilate under \( \otimes \) (i.e., \( 0 \otimes k = k \otimes 0 = 0, \forall k \in \mathbb{K} \)). A semiring can be specified in terms of a 5-tuple corresponding to \( (\mathbb{K}, \oplus, 0, \otimes, 1) \). The most common semiring is plus-times \( (\mathbb{R}, +, 0, \times, 1) \); other common examples of semirings are:

- min-plus \( (\mathbb{R} \cup \{\infty\}, \min, \infty, +, 0) \),
- max-plus \( (\mathbb{R} \cup \{-\infty\}, \max, -\infty, +, 0) \),
- min-product \( (\mathbb{R}^{\geq 0} \cup \{\infty\}, \min, \infty, \times, 1) \), and
- max-product \( (\mathbb{R}^{\geq 0}, \max, 0, \times, 1) \).

As shown later, our RMLP model utilizes the max-product semiring.

3.2 Weighted Finite State Automaton (WFSA)

WFSA [67]s are commonly used in natural language processing (NLP) to model text strings, and we use a simple example in that domain to provide an intuitive understanding of them. In Figure 1, circles represent states; a bold circle represents an initial state and a double circle represents a final state. A transition from one state to another is represented by a directed arc from the former to the latter. Each arc is labeled with an input symbol (e.g., 'early') and a weight (e.g., 0.6) that is associated with the transition.

Beginning from the initial state, we traverse contiguous arcs in the direction of their arrows, each time consuming the symbol associated with an arc, and accumulating its weight. No input is consumed when traversing an arc that is labeled with the empty symbol \( \epsilon \). In this manner, a WFSA provides a mapping from a sequence of symbols (e.g., words) to a sequence of (scalar) weights. A transition from one state to another is represented by a directed arc from the former to the latter. The label of a path is the concatenation of the labels of its constituent arcs, and the weight of a path is the product of the weights of the constituent arcs.

An input sequence \( x_1, \ldots, x_n \) is termed to be accepted or matched by a WFSA if there exists a successful path in the WFSA that is labeled \( x_1, \ldots, x_n \) (modulo the \( \epsilon \) symbol). For example, the WFSA in Figure 1 accepts “Awakening very early daily” and “Awakening early”, but not “Awakening too early” or “Awakening daily”.

In this WFSA, its initial state (state 0) signifies the condition in which the WFSA has not seen any words yet. At state 0, if the WFSA encounters the first word “Awakening” in its input string, its state transitions from state 0 to state 1, as indicated by the leftmost arc. State 1 represents the condition in which the WFSA has seen the word “Awakening”, followed by zero or more occurrences of the word “very”. Note that the self-loop at state 1, labeled with the word “very”, allows the WFSA to transition from state 1 back to state 1 upon encountering one or more instances of “very”. At state 1, when

\[ \text{very} / 1.2 \]

\[ \text{Awakening} / 1.0 \quad \text{early} / 0.6 \]

\[ \quad \text{daily} / 0.7 \]

\[ \epsilon / 0.2 \]

\[ \quad \text{state 0} \quad \text{state 1} \quad \text{state 2} \quad \text{state 3} \]

Fig. 1. An example WFSA.

ACM Trans. Manag. Inform. Syst., Vol. 0, No. 0, Article 0. Publication date: 2024.
the WFSA encounters “early” as the next word in its input, it transitions to state 2. Thus, state 2 represents the condition in which the WFSA has seen an input starting with the word “Awakening”, followed by zero or more occurrences of the word “very”, and then followed by the word “early”. At state 2, the WFSA can transition to state 3 in one of two ways: (a) when the input ends (using the arc labeled $\epsilon$), or (b) when it encounters the word “daily” as the last word in its input (using the arc labeled “daily”). Therefore, state 3 represents the condition in which the WFSA has encountered what it has seen at state 2 followed optionally by the word “daily”. In summary, the WFSA only accepts strings it deems valid, defined as those beginning with “Awakening”, followed by zero or more occurrences of “very”, followed by “early”, and immediately ending thereafter or ending with “daily”. Intuitively, the WFSA is matching input sequences to a pattern, and only accepts those that conform to the pattern.

The weight of a successful path is given by the product of the weights on the arcs along the path. The weight associated with an input sequence $x_1, \ldots, x_n$, also known as its score, is found by considering the weights of all successful paths labeled $x_1, \ldots, x_n$ (modulo the $\epsilon$ symbol) and picking the largest one. These scores indicate the WFSA’s preference for strings that it accepts, with higher scores indicating a stronger preference for the corresponding strings. In Figure 1, the WFSA only has one successful path for the input “Awakening early daily”, and the score of the input is equal to weight of the path, i.e., $1.0 \times 0.6 \times 0.7 = 0.42$. Similarly, the WFSA only has one successful path for the input “Awakening early”, and the score of the input is $1.0 \times 0.6 \times 0.2 = 0.12$. Consequently, the WFSA favors “Awakening early daily” over “Awakening early”.

Formally, a WFSA is associated with a semiring $(\mathbb{K}, \oplus, 0, \otimes, 1)$ that defines its addition and multiplication operators. A WFSA is defined as a 5-tuple $(V, Q, T, \pi, \eta)$, where $V$ is the input set, $Q$ is a set of states with size $|Q| = d$, $\pi \in \mathbb{K}^{1 \times d}$ is an initial weight row vector, $\eta \in \mathbb{K}^{d \times 1}$ is a final weight column vector, and $T : (V \cup \{\epsilon\}) \rightarrow \mathbb{K}^{d \times d}$ is a transition weight function. $T(\cdot)$ can be interpreted as a $d \times d$ matrix. (T need not be constrained to take a discrete symbol as input; as will be shown later in Equation 6, it can also accept a real-valued vector as input.)

Given a sequence of inputs $x = (x_1, \ldots, x_n) \in V^n$, the Forward algorithm [9] scores $x$ with respect to a WFSA. In the absence of $\epsilon$-transitions, Forward expresses the score $p_{span}(x)$ as a series of matrix multiplications:

$$p'_{span}(x) = \pi \left( \prod_{i=1}^{n} T(x_i) \right) \eta.$$  

(1)

Since an $\epsilon$-transition occurs without consuming an input symbol $x_i \in V$, to incorporate $\epsilon$-transitions, we can rewrite Equation 1 as

$$p_{span}(x) = \pi T(\epsilon)^* \left[ \prod_{i=1}^{n} (T(x_i)T(\epsilon)^*) \right] \eta$$

(2)

$$= ([\ldots ((\pi T(\epsilon)^*) T(x_1)) T(\epsilon)^* ] \ldots T(x_n)) T(\epsilon)^* \eta)$$

where $*$ refers to matrix asteration ($A^* := \sum_{j=0}^{\infty} A^j$). Equation 2 can be rewritten recursively as

$$h_0 = \pi T(\epsilon)^*$$

(3)

$$h_{t+1} = (h_t T(x_{t+1})) T(\epsilon)^*$$

(4)

$$p_{span}(x) = h_n \eta$$

(5)

### 3.3 Neural WFSA

Schwartz et al. (2018; [62]) created a neural version of WFSA by using a neural network to learn the transition weight function $T(\cdot)$. For tractability, they also restricted transitions from each state $i$ to the following 3 kinds.

(1) **Self-loop**: The transition consumes an input symbol and stays at the same state $i$.  

The equations are numbered slightly out of order to group related equations together later (e.g., in Theorem A.8).
(2) **Main path**: The transition consumes an input symbol and moves to state $i + 1$. (Moving to state $i + k$ where $k \neq 1$ is not allowed.)

(3) **$\epsilon$-transition**: The transition moves to state $i + 1$ without consuming an input symbol.

(Note that the self-loop transition and $\epsilon$-transition allow a WFSA to accept an input sequence that is respectively longer and shorter than its number of states.)

The transition weight function $T(\cdot)$ is defined as

$$
T(x)_{i,j} = \begin{cases} 
E(u_i \cdot x + a_i) & \text{if } j = i \text{ (self-loop)}, \\
E(w_i \cdot x + b_i) & \text{if } j = i + 1, \\
0 & \text{otherwise},
\end{cases}
$$

where $x$ is now an input real-valued vector, $T(x)_{i,j}$ is the element at $(i, j)$ position of matrix $T(x)$, $u_i$ and $w_i$ are vectors of parameters (weights), $a_i$ and $b_i$ are scalar parameters (biases), and $E$ is an encoding function. Because $x$ is a vector (rather than a discrete symbol), a neural WFSA can flexibly accept a vector of real-valued features at each time step. For $\epsilon$-transitions, $T(\epsilon)$ is defined as

$$
T(\epsilon)_{i,j} = \begin{cases} 
E(c_i) & \text{if } j = i + 1, \\
0 & \text{otherwise},
\end{cases}
$$

where $c_i$ is a scalar parameter.

In a neural WFSA, the addition and multiplication operators are also defined by a semiring.

When computing the score of an input sequence $p_{\text{span}}(x)$ in a vanilla WFSA using Equations 3, 5, and 8, the WFSA is typically constrained to match the entire input sequence from the start to the end. However, in natural language processing tasks, often only a subset of the input matters despite the overall length. (For instance, the example in Section 1 demonstrates that only words 14-16 signify Robert’s ability in Spanish.) To enable WFSA to flexibly match subsequences without starting from the sequence’s beginning, Equation 8 can be modified as shown in Equation 4.

$$
h_{t+1} = \max \left( \langle h_t, T(x_{t+1}) \rangle \right) T(\epsilon)^* \approx \max \left( \langle h_t, T(x_{t+1}) \rangle \right) max(I, T(\epsilon)), \ h_0. \tag{4}
$$

By including $h_0$ in the max function, Neural WFSA can begin the recurrence relation at time $t + 1$, and start matching a subsequence beginning at time $t+1$ if it gives a higher score. Note that $T(\epsilon)^*$ in both Equations 3 and 4 is approximated as $I + T(\epsilon)$.

Additionally, a subsequence matched by WFSA need not necessarily conclude with the last input in a sequence. Equation 5 can be adjusted as shown in Equation 9.

$$
p_{\text{span}}(x) = \max_{1 \leq t \leq n} h_t \eta. \tag{9}
$$

This modification allows the neural WFSA to conclude the recurrence relation at any time $1 \leq t \leq n$ if it provides a higher score by computing $p_{\text{span}}(x)$ using the maximum of $h_t \eta$ over the range $1 \leq t \leq n$.

Peng et al. (2018; [54]) defines the sets of strings accepted by neural WFSAs as **rational languages**. Since our model RMLP is built upon neural WFSAs, we mnemonically name it **rational** multi-layer perceptrons to reflect its ability to learn patterns that are rational languages.

### 4 Rational Multi-Layer Perceptrons (RMLP)

#### 4.1 The RMLP Model

To create our RMLP model with WFSAs, we use the max-product semiring $(\mathbb{R}_{\geq 0}, \max, 0, \times, 1)$. Unlike Equation 6 in which the transition weight function is a linear model of a real-valued feature vector, our RMLP uses a multi-layer perceptron (MLP) as the transition weight function (Equation 10) in

---

4Rational languages are analogous to **regular** languages that are accepted by unweighted finite state automata.
order to model a complex combination of features in a non-linear manner. This richer formulation of the transition weight function allows our RMLP model to subsume neural WFSAs as its special case, and we rigorously show this in Proposition A.11 of Appendix A.1.4.

\[
T(\pi)_{i,j} = \begin{cases} 
    \text{MLP}(x; u_i) & \text{if } j = i \text{ (self-loop)}, \\
    \text{MLP}(x; w_i) & \text{if } j = i + 1, \\
    0 & \text{otherwise,}
\end{cases}
\]

where \( \text{MLP}(x; \theta) \) is the output of an MLP with parameters \( \theta \) (the weights and biases of every layer of the MLP) given input \( x \). Because RMLP uses the max-product semiring which only works with non-negative numbers, we use the sigmoid function, \( \sigma(x) := \frac{1}{1+e^{-x}} \), as the non-linear activation function in the last layer to ensure that the output is non-negative.

In neural WFSA, a transition weight is calculated using \( E(w \cdot x + b) \) (Equation 6), which resembles a single neuron in a deep learning model operating on an input vector \( x \) (or a word embedding in the context of NLP). This allows neural WFSA to learn a soft pattern. In contrast, in the vanilla WFSA (Figure 1), the exact input word \textit{Awakening} is required to trigger the transition from state 0 to 1. However, with a soft pattern, the word needed for the transition is computed using the inner product between the weights \( w_0 \) and input \( x \) in \( [T(x)]_{0,1} = E(w_0 \cdot x + b_0) \). Other words with embedding vectors close to \textit{Awakening} such as \textit{Waking} can also result in a high transition weight, since the inner product \( w_0 \cdot x \) is a projection of vector \( x \) onto \( w_0 \). However, our work is in the domain of health risk prediction, where each element of the vector \( x \) represents a clinical feature such as blood pressure. Thus our aim is to enable transition weight functions that can effectively model non-linear interactions among such clinical features. To achieve this, we replace the computation of transition weights, which was originally based on \( E(w \cdot x + b) \), with multi-layer perceptrons (MLPs).

### 4.2 RMLP Generalizes MLP

To elucidate the structure of our RMLP model, we first show that a simple multi-layer perceptron (MLP) is equivalent to an RMLP with 2 states (0 and 1) and no \( \epsilon \)-transition (achieved by setting \( T(\epsilon)^* \) to the identity matrix). In such an RMLP, the transition weight functions (Equations 10 and 7), initial and final weight vectors are

\[
\begin{align*}
T(x) &= \begin{bmatrix} \text{MLP}(x; u_0) & \text{MLP}(x; w_0) \\ 0 & \text{MLP}(x; u_1) \end{bmatrix}, & T(\epsilon)^* &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, & \pi &= \begin{bmatrix} 1 & 0 \end{bmatrix}, \\
\eta &= \begin{bmatrix} 0 \\ 1 \end{bmatrix}.
\end{align*}
\]

Using Equation 2, we get the score for input \( x_1 \) (a real-valued vector) as

\[
p_{\text{span}}(x_1) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \text{MLP}(x_1; u_0) & \text{MLP}(x_1; w_0) \\ 0 & \text{MLP}(x_1; u_1) \end{bmatrix} = \text{MLP}(x_1; w_0),
\]

which is \textit{exactly} the equation of an MLP model. (NB: Because we are using the max-product semiring, all sum operations resulting from the matrix multiplications in the above expression must be replaced with the max operation. This also applies to the matrix multiplications in subsequent equations.)
This is equivalent to multiplying the outputs of the MLPs, where each MLP output corresponds to a transition for an input $x$. Long sequences such as $x, x, x$ can also be matched with this RMLP by using self-loop transitions.

Next we illustrate with concrete examples that an RMLP with more than 2 states is equivalent to the accumulation of the outputs of MLPs over individual inputs in a sequence. For simplicity, we focus on an RMLP with 3 states (depicted in Figure 2(a)) with no $\epsilon$-transitions (we omit the identity $T(\epsilon)^{\ast}$ matrix in the multiplications below to save space). For such an RMLP, we have

$$T(x) = \begin{bmatrix}
MLP(x; \mathbf{u}_0) & MLP(x; \mathbf{w}_0) & 0 \\
0 & MLP(x; \mathbf{u}_1) & MLP(x; \mathbf{w}_1) \\
0 & 0 & MLP(x; \mathbf{u}_2)
\end{bmatrix},
T(\epsilon)^{\ast} = \begin{bmatrix}1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1\end{bmatrix}, \pi = \begin{bmatrix}1 & 0 & 0 \end{bmatrix}, \eta = \begin{bmatrix}0 \\ 1 \end{bmatrix}.$$ (11)

As before, using Equation 2, we get the score for input sequence $x, x, x$ as

$$p_{span}(x, x, x) = \begin{bmatrix}MLP(x; \mathbf{u}_0) & MLP(x; \mathbf{w}_0) & 0 \\
0 & MLP(x; \mathbf{u}_1) & MLP(x; \mathbf{w}_1) \\
0 & 0 & MLP(x; \mathbf{u}_2)
\end{bmatrix}, T(\epsilon)^{\ast} = \begin{bmatrix}1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1\end{bmatrix}, \pi = \begin{bmatrix}1 & 0 & 0 \end{bmatrix}, \eta = \begin{bmatrix}0 \\ 1 \end{bmatrix}.$$ (12)

This is equivalent to multiplying the outputs of the MLPs, where each MLP output corresponds to a transition for an input $x$. Longer sequences such as $x, x, x$ can also be matched with this RMLP by using self-loop transitions.

$$p_{span}(x, x, x) = \max(MLP(x; \mathbf{w}_0)MLP(x; \mathbf{w}_1)MLP(x; \mathbf{u}_2), MLP(x; \mathbf{w}_0)MLP(x; \mathbf{u}_1)MLP(x; \mathbf{w}_1), MLP(x; \mathbf{u}_0)MLP(x; \mathbf{w}_0)MLP(x; \mathbf{w}_1)).$$ (13)
The score of sequence $x_1, x_2, x_3$ is given by the maximum among:

1. $MLP(x_1; w_0) \ MLP(x_2; w_1) \ MLP(x_3; u_2)$, obtained by transitioning from state 0 to 1 for input $x_1$, from state 1 to 2 for $x_2$, and self-looping at state 2 for $x_3$ (Figure 2(b));
2. $MLP(x_1; w_0) \ MLP(x_2; u_1) \ MLP(x_3; w_1)$, obtained by transitioning from state 0 to 1 for input $x_1$, self-looping at state 1 for $x_2$, and transitioning from state 1 to state 2 for $x_3$ (Figure 2(c));
3. $MLP(x_1; u_0) \ MLP(x_2; w_0) \ MLP(x_3; w_1)$, obtained by self-looping at state 0 for input $x_1$, transitioning from state 0 to 1 for $x_2$, and transitioning from state 1 to 2 for $x_3$ (Figure 2(d)).

Each of these three cases involves the multiplication of MLPs, with each function applied over individual inputs in a sequence and for different transitions. This highlights how RMLP serves as a generalization of MLPs to handle sequential data.

4.3 RMLP for Predicting Diabetic Ketoacidosis

To build intuition about how our RMLP model works, we provide another example in Figure 3(a), which illustrates a 3-state RMLP for predicting diabetic ketoacidosis, a medical condition characterized by excessive urination followed by vomiting. This example is intentionally simplified to emphasize the core concept of our RMLP; it does not, however, capture our model’s capacity to learn intricate patterns.

The input to the RMLP is a sequence of clinical features of a patient at three consecutive time steps, denoted as $(x_1, x_2, x_3)$, with each $x_i$ being a multi-dimensional feature vector. Each element in the vector represents a specific clinical feature of the patient. For instance, $x_i = (\text{UrineVolume}_i, \text{Vomiting}_i)$ represents the urine volume and the number of vomiting episodes at timestep $i$ for the patient.

In the RMLP model, the initial state (state 0) represents the condition that the model has not observed a sufficiently large urination volume in the feature vectors of a patient. The model continues to self-loop in state 0 until it recognizes a significant urine volume of the patient. Once the RMLP identifies a considerably large urination volume in the feature vector $x_1$ at a particular timestep $i$, it transitions into state 1 (using the arc labeled $b$). The MLP on that edge accepts $x_i$ as input and uses its urination volume to output a weight for the edge. The weight reflects the severity of the clinical observation, with a larger weight corresponding to a greater severity. (An arbitrarily small weight then corresponds to the condition that only a small urination volume is observed.) State 1 represents the condition that the model has observed a substantial urination amount, and the model continues to remain in that state (via the self-loop labeled $c$) as long as it does not observe any noticeably frequent vomiting. Upon perceiving a notably large number of vomiting episodes within the feature vector $x_j$ of the patient at timestep $j$, the model transitions (via the arc labeled $d$) to state 2. The MLP on that edge accepts $x_j$ as input and uses its number of vomiting episodes to output a weight for the edge that captures the seriousness of the clinical observation. State 2 represents the condition that the model has observed the significant number of vomiting episodes after detecting the substantial urination volume (with other possible intervening observations between them). In state 2, the model either ends in that state, or self-loops (via the arc labeled $e$) and remains in that state. Note that the indices of $x_i$ and $x_j$ in the prior exposition are such that $i < j$. Thus the RMLP model captures the clinically significant temporally ordering in which vomiting occurs after excessive urination with the (joint) severity of those clinical events captured by the product of the weights of the path from state 0 to state 2. (This is in contrast to the attention mechanism used in healthcare predictive tasks, which captures features without necessarily modeling or enforcing their temporal ordering.) In Figures 3(b), (c), (d) and (e), we demonstrate the outcomes of applying the example RMLP model to four patients with the feature.

---

5The ordering of the symptoms is important.
vector sequences \((x_1, x_2, x_3), (y_1, y_2, y_3), (r_1, r_2, r_3)\) and \((z_1, z_2, z_3)\) respectively. The table below each figure shows the output values of the MLPs associated with the horizontal arcs for the specified feature vectors. In each figure, we show the best path that the RMLP has matched to each input sequence, with every arc labeled with the feature vector to which it is matched.

In Figure 3(b), the model uses Equation 13 to find the best path with the highest value among all possible paths for the input sequence \((x_1, x_2, x_3)\). Equation 13 considers all possible ways of matching the input sequence \((x_1, x_2, x_3)\) to the arcs of Figure 3(a), resulting in:

\[
 p_{\text{span}}(x_1, x_2, x_3) = \max(MLP(x_1; w_0)MLP(x_2; w_1), \\
 MLP(x_1; w_0)MLP(x_3; w_1), \\
 MLP(x_2; w_0)MLP(x_3; w_1)).
\]

Replacing the outputs of the MLPs with the values in the table of Figure 3(b), we obtain the following:

\[
 p_{\text{span}}(x) = \max(0.09 \times 0.03, \\
 0.09 \times 0.8, \\
 0.9 \times 0.8).
\]
Consequently, the RMLP model obtains \( p_{\text{span}}(x_1, x_2, x_3) = MLP(x_2; w_0) \cdot MLP(x_3; w_1) \) that corresponds to the path shown in the figure. In deriving that best path, the RMLP has matched \( x_1 \) to the self-loop of state 0. In addition, \( MLP(b; w_0) \) on the left horizontal edge of Figure 3(a) favors \( x_2 \) over \( x_1 \) and \( x_3 \) because \( x_2 \) contains the highest amount of urine (2,500 milliliters). Similarly, \( MLP(d; w_1) \) on the right horizontal edge of Figure 3(a) favors \( x_3 \) over \( x_1 \) and \( x_2 \) because \( x_3 \) contains the most number of vomiting episodes (3 times). The patient in Figure 3(b) exhibits characteristics indicative of diabetic ketoacidosis, namely excessive urination followed by frequent vomiting, resulting in a score of \( p_{\text{span}}(x_1, x_2, x_3) = 0.9 \times 0.8 = 0.72 \) (the 1.0 weight of the self-loop is omitted from the product because it does change the score).

In contrast, in Figure 3(e), the patient exhibits normal urination levels and no instances of vomiting at all timesteps. Although, the RMLP model still matches the input \((z_1, z_2, z_3)\) as shown in the figure, it does so by giving the best path a score of \( p_{\text{span}}(z_1, z_2, z_3) = 0.12 \times 0.02 = 0.024 \). The score is significantly lower than that in Figure 3(b), capturing the fact that the patient in Figure 3(e) is a lot less likely to have diabetic ketoacidosis.

In Figures 3(c) and (d), we illustrate the flexibility that self-loops provide. In Figure 3(c), the feature vector \( y_1 \) is matched to the left horizontal arc because it contains the highest amount of urine (2,500 milliliters) when compared to \( y_2 \) and \( y_3 \). However, there is no vomiting in feature vector \( y_2 \) in the subsequent step. Consequently, \( y_2 \) is matched to the self-loop at state 1 so that \( y_3 \), which contains a large number of vomiting episodes, can be matched to the right horizontal arc. Thus, the self-loop allows for input sequences in which clinically significant events can be ordered temporally without a fixed time gap between them.

Similarly, in Figure 3(d), the feature vectors \( r_1 \) and \( r_2 \) are matched to the first and second horizontal arcs respectively because \( r_1 \) contains the highest urine volume (2,500 milliliters) and \( r_2 \) has the largest number of vomiting episodes. The last feature vector \( r_3 \), is matched to the self-loop at the final state. The self-loop at the final state allows for input sequences in which the last feature vector is not the most significant event. For both Figures 3(c) and (d), the resulting scores for the best paths are \( p_{\text{span}}(y_1, y_2, y_3) = 0.9 \times 0.8 = 0.72 \) and \( p_{\text{span}}(r_1, r_2, r_3) = 0.9 \times 0.8 = 0.72 \) respectively. These high scores are equal to that in Figure 3(b), capturing the fact that their associated patients are as likely as that in Figure 3(b) to have diabetic ketoacidosis.

In examples Figure 3(b), (c), (d) and (e), we have shown that the score of an input sequence \((x_1, x_2, x_3)\) is determined by the product \( MLP(x_i_1; w_0) \cdot MLP(x_i_2; w_1) \), where (i) \((i_1, i_2)\) is the subsequence of \((1,2,3)\) that yields the path with the highest value, (ii) \( MLP(x_i_1; w_0) \) accounts for the volume of urination, and (iii) \( MLP(x_i_2; w_1) \) accounts for the number of vomiting episodes.

### 4.4 Importance of \( \epsilon \)-transitions in RMLP

Till now we have disallowed \( \epsilon \)-transitions. Recall from Section 3.3 that \( T(\epsilon)^* \approx I + T(\epsilon) = \max(I, T(\epsilon)) \) (RMLP uses the max-product semiring that uses max as its addition operator). Thus, \( T(\epsilon)^* \) is as follows (the resulting RMLP is shown in Figure 4(a)):

\[
T(\epsilon)^* = \begin{bmatrix} 1 & \sigma(c_0) & 0 \\ 0 & 1 & \sigma(c_1) \\ 0 & 0 & 1 \end{bmatrix}.
\]

We shall see below that \( \epsilon \)-transitions allow RMLP to accept a sequence \((x_1)\) whose length is shorter than the number of main path transitions needed to reach the end state (2 in this case). The
score of $x_1$ is

\[
P_{\text{span}}(x_1) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & \sigma(c_0) & 0 \\ 0 & 1 & \sigma(c_1) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \text{MLP}(x_1; w_0) & \text{MLP}(x_1; w_0) & \text{MLP}(x_1; w_0) \\ \text{MLP}(x_1; u_1) & 0 & \text{MLP}(x_1; u_1) \\ 0 & 0 & \text{MLP}(x_1; u_2) \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \sigma(c_1) \\ 0 & 0 & \sigma(c_1) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\]

\[
\max(\text{MLP}(x_1; w_0)\sigma(c_1), \sigma(c_0)\text{MLP}(x_1; u_1)\sigma(c_1), \sigma(c_0)\text{MLP}(x_1; w_1)).
\]

This shows that the sequence with length 1 can be accepted by the RMLP with 3 states and its score is equal to the maximum of

1. $\text{MLP}(x_1; w_0)\sigma(c_1)$, obtained by transitioning from state 0 to 1 by input $x_1$, and transitioning from state 1 to 2 by an $\epsilon$-transition (Figure 4(b)).
2. $\sigma(c_0)\text{MLP}(x_1; u_1)\sigma(c_1)$, obtained by transitioning from state 0 to 1 by an $\epsilon$-transition, self-looping at state 1 by $x_1$, and transitioning from state 1 to 2 by an $\epsilon$-transition (Figure 4(c)).
3. $\sigma(c_0)\text{MLP}(x_1; w_1)$, obtained by transitioning from state 0 to 1 by an $\epsilon$-transition, and transitioning from state 1 to 2 by input $x_1$ (Figure 4(d)).

To underscore the significance of $\epsilon$-transitions, particularly in the healthcare domain’s application of our RMLP, it is crucial to note that the number of states is a predetermined hyperparameter, not directly learned from the data. With an $\epsilon$-transition, the RMLP can transition between states without consuming input. For instance, let us consider a 3-state RMLP that uses information about whether a patient is a smoker (such an RMLP may be used, for example, to predict the patient’s risk of having a stroke [63]). However, for this particular task, a 2-state RMLP would suffice. State 0 represents the initial condition where the RMLP has not observed the patient’s smoking history, while state 1 indicates that the RMLP has detected the smoking history. By specifying a 3-state RMLP for this task, the RMLP could optimize itself by utilizing an $\epsilon$-transition for the transition from state 0 to state 1. For example, given sequential input $x_1, x_2$, where $x_1 = (\text{Smoking}_i)$ with $x_1 = (0)$ and $x_2 = (1)$ (where Smoking$_i = 1$ indicates smoking at timestep $i$), transitioning from state 0 to 1 can be achieved using an $\epsilon$-transition, while transitioning from state 1 to 2 is done by $x_2$. Consequently, a 3-state RMLP can be used for this task for which a 2-state RMLP is sufficient.

4.5 Flexible Subsequence Matching in RMLP

Instead of requiring an entire input sequence to be matched (like in a traditional WFSA), an RMLP is able to match a subsequence in its input. In practice, we allow our RMLP to flexibly match subsequences that do not start at the beginning of the input sequence. However, due to the importance of the most recent clinical information located at the end of the input sequence, we
refrain from allowing our RMLP to match subsequences that do not conclude with the last segment of the input sequence. To this end, we use Equations 3, 4, and 5 to derive $p_{span}(x)$ for our RMLP model.

### 4.6 RMLP Generalizes Static Models to Sequential Data

In our RMLP model, its component MLPs amalgamate scores reflecting important clinical events in sequential input data (e.g., excessive urination or vomiting) by multiplying them together. In this way, our RMLP model can be seen as a generalization of MLPs for sequential data.

Formally, we denote this generalization as GSMSD(Multi-Layer Perceptron) in Definition A.4, and prove the following proposition in Appendix A.1.3.

**Proposition A.10.** The function $p_{span}'(x)$, as defined by the following Equations 3, 4, 5, 22 and 23 using the max-product semiring $(\mathbb{R}_{\geq 0}, \max, 0, \times, 1)$, is a GSMSD(Multi-Layer Perceptron) function.

\[
\begin{align*}
    h_0 &= \pi T(\epsilon)^* = \pi \max(I, T(\epsilon)), \\
    h_{t+1} &= \max (\{h_t T(x_{t+1}) T(\epsilon)^*, h_0\}, \\
    &= \max (\{h_t T(x_{t+1}) \max(I, T(\epsilon)), h_0\}), \\
    p_{span}'(x) &= h_n \eta,
\end{align*}
\]

where $\pi T(\epsilon)^*$ is approximated as $\pi \max(I, T(\epsilon))$ (as mentioned in Section 4.1), $\pi = [1 0 \ldots 0] \in \mathbb{R}^{1 \times D}$ is an initial weight vector, $\eta = [0 \ldots 0 1]^T \in \mathbb{R}^{D \times 1}$ is a final weight vector, and $D$ is the number of states. The transition weight function $T(\cdot)$ is given by the following equations.

\[
[T(x)]_{i,j} = \begin{cases} 
    \text{MLP}(x; u_i) & \text{if } j = i \text{ (self-loop)}, \\
    \text{MLP}(x; w_i) & \text{if } j = i + 1, \\
    0 & \text{otherwise}, 
\end{cases}
\]

\[
[T(\epsilon)]_{i,j} = \begin{cases} 
    \sigma(c_i) & \text{if } j = i + 1, \\
    0 & \text{otherwise}, 
\end{cases}
\]

where MLP($x; \theta$) is the output of an MLP given input $x$ and parameters $\theta$ (i.e., the weights and biases of every layer in the MLP), and the sigmoid function $\sigma(\cdot)$ is used as the non-linear activation in the last layer.

By utilizing MLPs as transition weight functions (Equation 10), our RMLP model adapts MLPs to handle sequential data. We use MLPs as components in our RMLP model because of their capacity to learn intricate nonlinear features that are useful for our healthcare tasks. In Theorem A.8 (Appendix A.1.1), we formally show how any non-negative static model (such as MLPs) can be extended to handle sequential data. In addition, in Proposition A.11 (Appendix A.1.4), we show that RMLP subsumes neural WFSA as a special case. This seems intuitively plausible because RMLP’s transition weight functions are modeled as MLPs, which are more expressive than the logistic regression function that is used in neural WFSA’s transition weight functions.

Thus, RMLP generalizes both MLP and neural WFSA. It combines the ability of MLP to learn complex non-linear features with WFSA’s ability to represent interpretable temporal information. In this way, RMLP is able to capture correlations among features across timesteps in an input sequence that attention mechanisms cannot represent.

Although we have proposed RMLP as our specific model, we wish to reiterate that Theorem A.8 (Appendix A.1.1) demonstrates that our approach is general enough to allow any non-negative static model to be adapted to handle sequential data.
4.7 Choice of Semiring in RMLP
An aspect that requires explanation is the rationale behind selecting the max-product semiring for our RMLP model, i.e., choosing the max operation as the \(\oplus\) (addition) operator and the product operation as the \(\otimes\) (multiplication) operator. As demonstrated by the examples in Figure 3, the max function enables us to choose the best path among all available paths that results in the highest score. This serves our goal of pinpointing the most relevant subsequence of clinical events for a predictive task. Once we have opted for the max function as our addition operator, the decision between the max-plus and max-product semirings is based on an evaluation of their respective predictive performance on a validation dataset. This assessment helps us determine which semiring is better suited for our specific tasks.

In summary, our RMLP model uses WFSAs to learn temporal patterns within sequential input that are indicative of the occurrence of a target clinical event. In RMLP, the \(p_{span}\) value (Equation 13) indicates how closely a sequential input is aligned with the learned pattern, and an MLP is employed at each timestep to measure the suitability of the input vector at that timestep to effect a state transition in the pattern.

Rather than using only a single WFSA, our RMLP model can include multiple WFSAs, with the number of WFSAs and their lengths (i.e., number of states) specified as hyperparameters. We term each WFSA in our RMLP model as a pattern because a WFSA represents a sequential regularity in the input. We use the terms WFSA and pattern synonymously. (Figure 5 illustrates an example RMLP model with 2 patterns, one of length 3 and another of length 4.) We stack the \(p_{span}(x) = b_n\eta\) scores (Equations 4 and 5) of all patterns into a vector (where \(n\) is the length of an input sequence), and pass that vector as input to a final multi-layer perceptron.

To learn the parameters of our RMLP model, i.e., the weights and biases of its constituent WFSAs (the parameters in Equations 7 and 10) and the parameters of the final MLP, we use cross-entropy as our objective function, and minimize it by training the RMLP model end-to-end via backpropagation. To reduce the number of patterns used, we group all the weights and biases associated with each pattern, and penalize our model using the \(L^2\) norm on the weights and biases.

![Diagram of RMLP model](image)

*Fig. 5. A schematic of our RMLP model, with 2 patterns (one of length 4 and one of length 3), taking a sequence of clinical information about a patient as input. The transition from one rectangular block of states to another is governed by MLPs. At the end of the input sequence, the score from each pattern (Equation 4) are stacked and fed into a final multi-layer perceptron to make predictions.*
4.8 Time Complexity

To evaluate the time complexity of training our RMLP model, we examine the computations in the forward pass of RMLP. The forward pass computations can be expressed through Equations 3, 4, and 5. In these equations, we denote the time complexity for calculating $T(x)$ as $O(dW_f^{MLP})$, where $W$ represents the number of WFSAs used in RMLP, and $d$ represents the maximum number of states among the WFSAs within RMLP. Additionally, $f_{MLP}$ represents the time complexity of an MLP for each transition weight.

Consequently, when calculating $p_{span}(x)$ using Equations 3, 4, and 5, its time complexity amounts to $O(dWn) + O(dW_f^{MLP})$, where $n$ represents the sequence length of $x$. The additional time complexity of $O(dWn)$ arises because the computation in Equation 4 is performed recursively for $n$ iterations, and this repetition occurs for each WFSA. Each iteration involves matrix multiplication of diagonal matrices $T(x_{t+1})$ and $T(\epsilon)^*$. Please note that both matrices have only one row of upper diagonal elements, as we only allow transitions from state $j$ to $j + 1$ for transitions that are not self-loops.

With $B$ samples, the time complexity becomes $O(BdWn) + O(BdW_f^{MLP})$.

$$p_{span}(x) = h_n \eta$$

$$h_0 = \pi T(\epsilon)^*$$

$$h_{t+1} = \max\left( \left( h_t T(x_{t+1}) \right) \left( \max(I, T(\epsilon)) \right) , h_0 \right)$$

$$h_0 = \pi T(\epsilon)^*$$

Fig. 6. Computational Graph of WFSA

We omit the time complexity of the final MLP calculation in this context, as it can be derived from the time complexity described in [49] and is not the primary focus of our discussion. Moreover, in practical scenarios, the final MLP is typically small in size especially when compared to the WFSAs within the RMLP model. Consequently, the dominant factor in the time complexity is the calculation involving the WFSA components.

To update the model’s parameters using gradient descent, we must compute the derivatives of the training loss with respect to the model’s parameters, including $c_s$, $w_s$, and $u_s$ for $s = 0, \ldots, d - 1$, where $d$ is the number of states of a WFSA. These derivatives can be obtained through the chain rule, and the most time-consuming part involves computing the derivatives of the WFSA’s output, denoted as $p_{span}$, with respect to its parameters: $\frac{\partial p_{span}}{\partial u_s}$, $\frac{\partial p_{span}}{\partial w_s}$, and $\frac{\partial p_{span}}{\partial c_s}$ for each WFSA.
To understand the time complexity of computing derivatives for a single WFSA, please refer to Figure 6. This figure illustrates the relationships between variables by connecting them with lines, with higher-level variables being derived from lower-level ones. Note that only variables dependent on the model’s parameters are shown.

For example, considering Equation 5, we see that \( p_{\text{span}}(x) \) is computed from \([h_n]_s\) for \( s = 0, \ldots, d - 1 \), where \([h_n]_s\) represents the \( s^{th} \) element of the vector \( h_n \). We exclude \( \eta \) since it is a constant unrelated to RMLP’s parameters.

The graph helps us quantify the number of operations required to compute \( \partial p_{\text{span}} / \partial u_s \), \( \partial p_{\text{span}} / \partial c_s \) and \( \partial p_{\text{span}} / \partial c_s \). For instance, to determine the operations needed for \( \partial p_{\text{span}} / \partial u_s \), we start by finding the partial derivatives of \( p_{\text{span}} \) with respect to \([h_n]_s\) for \( s = 0, \ldots, d - 1 \), which are the variables connected to \( p_{\text{span}} \) in the graph. Then, for each \([h_n]_s\) (colored in orange), we must compute the derivative of \([h_n]_s\) with respect to the variables linked to it, also highlighted in orange.

This orange part of the graph can be constructed based on Equation 4 at time sequence \( n \), which corresponds to the length of input \( x \). Equation 4 defines the relationship between the current step \( n \) and the previous step \( n - 1 \), involving the model’s parameters outlined in the 3rd row. By iteratively applying Equation 4, the graph extends until it reaches the initial step in the input sequence, denoted as \( 0 \).

To compute \( \partial p_{\text{span}} / \partial u_s \), we can leverage the graph to determine all the partial derivatives of the top variable with respect to the lower variables linked to it, until we reach the lowest level where the graph’s leaves are \( u_s \). By applying the chain rule, multiplying these partial derivatives together results in \( \partial p_{\text{span}} / \partial u_s \). Consequently, the number of operations required for finding the derivatives equals the number of links in the graph.

The orange portion of the graph pertains to a single step within a sequence of length \( n \) and a single state of a single WFSA. Therefore, the time complexity amounts to \( O(dWn) \) for a single training sample. When considering all \( B \) samples, the time complexity of this gradient calculation becomes \( O(BdWn) \).

Taking into account both the computation time of \( p_{\text{span}} \) in RMLP (forward pass), which is \( O(dWn) + O(dWf_{MLP}) \), and the calculation of its derivatives (backward pass), the overall training time complexity can be expressed as \( O(dW(n + f_{MLP})T) \), where \( T \) represents the number of training epochs.

5 Experiments

5.1 Datasets

We use two real-world clinical datasets (one public and one private) for six predictive tasks. The clinical events to be predicted range from short-term (within 24 hours) to long-term (within 1 year). Table 1 provides details about the data sizes, and the train/validation/test splits for each task.

5.1.1 MIMIC-III Dataset. We use the publicly available Medical Information Mart for Intensive Care (MIMIC-III) dataset [35]. The dataset contains longitudinal information about clinical events and medical outcomes of 33,798 unique patients with a total of 42,276 ICU stays. Harutyunyan et al. (2019; [30]) used MIMIC-III to prepare data for several benchmark tasks. We adopt that paper’s data preparation methodology, and focus on the benchmark tasks of in-hospital mortality prediction and decompensation prediction. To represent the clinical information about a patient, we create a 76-dimensional vector for every hour of his ICU stay, with each element in the vector corresponding to a clinical feature about the patient.
In-hospital Mortality Prediction. For this task, we have to predict whether a patient dies during his ICU stay or survives to be discharged from ICU. The problem is one of binary classification. The input data consists of clinical events associated with each patient within the first 48 hours of his ICU stay. The exclusion of ICU stays that are less than 48 hours reduced the data size from 42,276 ICU stays to 21,139.

 Decompensation Prediction. For this task, we have to predict whether a patient dies within the next 24 hours from every hour of his ICU stay. Because every hour of an ICU stay generates an example, a long stay produces many examples. Altogether there are 3,431,622 examples. This task is also one of binary classification.

5.1.2 Diabetes Comorbidities Dataset (DCD). DCD is a proprietary dataset from a government hospital. This dataset contains about 190,000 de-identified patients who are afflicted with type 2 diabetes, spans a duration of 8 years (2011-2018), and covers a comprehensive collection of patient-specific longitudinal clinical information (e.g., demographic details, physiological measurements, diagnoses, medications, laboratory tests, surgical procedures, and visit dates). We preprocess the clinical data into a 57,153-dimensional vector for each 10-day interval in every patient’s record. Each element in the vector corresponds to a clinical feature describing its associated patient. Because the data size is too large for our model and baselines to fit into computer memory, we compress the 57,153-dimensional input data down to 610-dimensional input vectors using autoencoders [61]. For this dataset, we focus on the tasks of predicting the onset of four diabetic comorbidities, viz., eye complications, ischemic stroke, hemorrhagic stroke, and transient ischemic attack. For a comorbidity, each 10-day interval (including all information preceding it) constitutes an example, and it is labeled as positive if its associated patient is diagnosed with the comorbidity within a year from the end date of that interval (otherwise, it is labeled as negative).

 Eye Complications. Diabetes is a leading cause of eye complications (e.g., diabetic retinopathy and macular edema) that could potentially result in permanent blindness. The vast majority of patients who develop such eye complications do not exhibit symptoms until the very late stages, by which time it is usually too late for effective treatment. Being able to accurately predict the onset of eye complications has the potential of identifying at-risk patients for early remedial treatment.

 Ischemic Stroke. Diabetes is a well-established risk factor for ischemic stroke [14], which in turn, can result in mortality and other adverse outcomes such as pneumonia. Being able to predict the future onset of ischemic stroke has the potential of identifying at-risk patients for whom preemptive medical measures could be administered.

 Hemorrhagic Stroke. Diabetes is associated with a higher incidence of hemorrhagic stroke, which is a leading cause of mortality and disability [71]. Like for ischemic stroke, an accurate prediction of the onset of hemorrhagic stroke helps physicians take early preventive measures for high-risk patients.

 Transient Ischemic Attack (TIA). The symptoms of a transient ischemic attack typically do not last long, but an attack can cause persistent cognitive impairment [72]. Through accurate prediction of TIAs, we can potentially identify at-risk patients for mitigatory medicinal interventions.

5.2 RMLP Model and Baselines

We set the hyperparameters of our RMLP model (and those of the baselines) by tuning on validation data, and the hyperparameters of our RMLP model are summarized in the Appendix A.2.

We compare our RMLP model against logistic regression, LIME [60], RETAIN [15], and Adacare [46]. Following previous literature [30], logistic regression is the natural baseline to choose as a static
Table 1. Data sizes and splits. +ve: the number of positive examples, -ve: the number of negative examples

<table>
<thead>
<tr>
<th>Task</th>
<th>Train (+ve; -ve)</th>
<th>Validation (+ve; -ve)</th>
<th>Test (+ve; -ve)</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-hospital Mortality</td>
<td>14,681 (1,987; 12,694)</td>
<td>3,222 (436; 2,786)</td>
<td>3,236 (374; 2,862)</td>
</tr>
<tr>
<td>(MIMIC-III)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decompensation</td>
<td>2,377,768 (49,261; 2,328,507)</td>
<td>530,646 (11,752; 518,894)</td>
<td>523,208 (9,683; 513,525)</td>
</tr>
<tr>
<td>(MIMIC-III)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eye Complications</td>
<td>126,681 (16,082; 110,599)</td>
<td>27,154 (3,462; 23,692)</td>
<td>27,159 (3,455; 23,704)</td>
</tr>
<tr>
<td>(DCD)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ischemic Stroke</td>
<td>125,545 (7,123; 118,422)</td>
<td>26,905 (1,524; 25,381)</td>
<td>26,912 (1,541; 25,371)</td>
</tr>
<tr>
<td>(DCD)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hemorrhagic Stroke</td>
<td>129,479 (1,526; 127,953)</td>
<td>27,732 (327; 27,405)</td>
<td>27,736 (325; 27,411)</td>
</tr>
<tr>
<td>(DCD)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIA</td>
<td>129,314 (2,098; 127,216)</td>
<td>27,707 (450; 27,257)</td>
<td>27,719 (447; 27,272)</td>
</tr>
<tr>
<td>(DCD)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

model against which we evaluate our model. (For a static model such as logistic regression, we have to 'flatten' the longitudinal EHR data by computing summary statistics for each feature, e.g., minimum, maximum, average, and standard deviation.) Another baseline is LIME. It is selected because it is one of the commonly utilized approaches for interpretability. It is a post-hoc technique applied to post-process predictions derived from a selected trained model, specifically chosen as an MLP in our case. RETAIN and Adacare (see Section 2) are chosen as baselines because they are state-of-the-art systems that model longitudinal healthcare data, and offer some degree of interpretability.

5.3 Results

Table 2 reports the performances of our RMLP model and the baselines on the six aforementioned tasks. We compare the systems using four metrics on test data: area under the precision-recall curve (AUPRC), area under the receiver operating characteristic curve (AUROC), log-likelihood (LL), and positive predictive value (PPV). For all metrics, the larger the number, the better the performance.

LL is the average over the log-likelihoods of test examples in a dataset. AUPRC and AUROC are computed by varying the threshold log-likelihood above which a test example is predicted to be true. PPV measures the proportion of top-$n$ positive predictions that are truly positive, where $n$ is a user-defined positive integer.

The advantage of using LL is that it directly measures the quality of the probability estimates produced (each system predicts the probability that a test example is true). AUROC is not particularly suitable for our imbalanced datasets (in which there are a lot more negative examples than positive ones) because it is insensitive to changes in positive predictions in such skewed regimes [27]. However, we include it because our baseline systems had used it in their previous evaluations. The advantage of AUPRC is that it is sensitive to the predictions on the relatively small number of positive test examples, and is less sensitive to the large number of true negatives. For highly skewed datasets (like ours), AUPRC is the metric of choice [21, 27], and is therefore our key metric (similar to Ma et al. (2020; [46])). In computing PPV, we are interested in the proportion of top-$n$ predicted high-risk patients who are truly positive. The PPV measure is particularly useful in situations where medical interventions or treatments are costly, and have to be limited to a restricted number.
of patients at highest risk. PPV reflects how successful we are in allocating such scarce resources to patients who need them. When computing PPV for each prediction task, we set $n$ to three different values depending on the number of available positive test examples, and denote the values as PPV-Small, PPV-Medium, and PPV-Large. In our experiments, we set PPV-Small/PPV-Medium/PPV-Large to 100/300/400 (in-hospital mortality), 1000/5000/8000 (decompensation), 400/1400/2400 (eye complication), 400/1000/1400 (ischemic stroke), 100/300/400 (hemorrhagic stroke), and 100/300/400 (transient ischemic attack).
Table 2. Empirical comparisons of RMLP against baseline systems. * and ** indicate statistical significance at the 95% and 99% level respectively.

<table>
<thead>
<tr>
<th>MIMIC-III</th>
<th>In-hospital Mortality Prediction</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Logistic Regression</td>
<td>LIME</td>
<td>RETAIN</td>
<td>Adacare</td>
<td>RMLP (Ours)</td>
</tr>
<tr>
<td>AUPRC</td>
<td>0.4720</td>
<td>0.3892</td>
<td>0.4514</td>
<td>0.4729</td>
<td><strong>0.5047</strong></td>
</tr>
<tr>
<td>AUROC</td>
<td>0.8450</td>
<td>0.7599</td>
<td>0.8467</td>
<td>0.8521</td>
<td><strong>0.8544</strong></td>
</tr>
<tr>
<td>LL</td>
<td>-0.2680</td>
<td>-0.3876</td>
<td>-0.2692</td>
<td>-0.2668</td>
<td><strong>-0.2606</strong></td>
</tr>
<tr>
<td>PPV-Small</td>
<td>0.7300</td>
<td>0.6200</td>
<td>0.6200</td>
<td>0.7200</td>
<td><strong>0.7500</strong></td>
</tr>
<tr>
<td>PPV-Medium</td>
<td>0.5267</td>
<td>0.4767</td>
<td>0.5000</td>
<td>0.5133</td>
<td><strong>0.5467</strong></td>
</tr>
<tr>
<td>PPV-Large</td>
<td>0.4575</td>
<td>0.4275</td>
<td>0.4425</td>
<td>0.4625</td>
<td><strong>0.4650</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MIMIC-III</th>
<th>Decompensation Prediction</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Logistic Regression</td>
<td>LIME</td>
<td>RETAIN</td>
<td>Adacare</td>
<td>RMLP (Ours)</td>
</tr>
<tr>
<td>AUPRC</td>
<td>0.2132</td>
<td>0.1285</td>
<td>0.2597</td>
<td>0.3029</td>
<td><strong>0.3248</strong></td>
</tr>
<tr>
<td>AUROC</td>
<td>0.8700</td>
<td>0.6336</td>
<td>0.8764</td>
<td>0.9009</td>
<td>0.8999</td>
</tr>
<tr>
<td>LL</td>
<td>-0.0708</td>
<td>-0.1317</td>
<td>-0.0692</td>
<td>-0.0686</td>
<td><strong>-0.0711</strong></td>
</tr>
<tr>
<td>PPV-Small</td>
<td>0.6500</td>
<td>0.6510</td>
<td>0.7760</td>
<td>0.7450</td>
<td><strong>0.8220</strong></td>
</tr>
<tr>
<td>PPV-Medium</td>
<td>0.3856</td>
<td>0.2776</td>
<td>0.4042</td>
<td>0.4634</td>
<td><strong>0.5046</strong></td>
</tr>
<tr>
<td>PPV-Large</td>
<td>0.3084</td>
<td>0.2004</td>
<td>0.3399</td>
<td>0.3798</td>
<td><strong>0.3951</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DCD</th>
<th>Eye Complications Prediction</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Logistic Regression</td>
<td>LIME</td>
<td>RETAIN</td>
<td>Adacare</td>
<td>RMLP (Ours)</td>
</tr>
<tr>
<td>AUPRC</td>
<td>0.6031</td>
<td>0.5238</td>
<td>0.4719</td>
<td>0.6627</td>
<td><strong>0.7422</strong></td>
</tr>
<tr>
<td>AUROC</td>
<td>0.8879</td>
<td>0.8194</td>
<td>0.8325</td>
<td>0.8740</td>
<td><strong>0.9165</strong></td>
</tr>
<tr>
<td>LL</td>
<td>-0.4161</td>
<td>-0.3892</td>
<td>-0.3158</td>
<td>-0.2468</td>
<td><strong>-0.2136</strong></td>
</tr>
<tr>
<td>PPV-Small</td>
<td>0.8100</td>
<td>0.7250</td>
<td>0.7275</td>
<td>1.0000</td>
<td><strong>0.9950</strong></td>
</tr>
<tr>
<td>PPV-Medium</td>
<td>0.7721</td>
<td>0.7150</td>
<td>0.6171</td>
<td>0.8729</td>
<td><strong>0.8957</strong></td>
</tr>
<tr>
<td>PPV-Large</td>
<td>0.6833</td>
<td>0.6521</td>
<td>0.5575</td>
<td>0.7050</td>
<td><strong>0.7938</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DCD</th>
<th>Ischemic Stroke Prediction</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Logistic Regression</td>
<td>LIME</td>
<td>RETAIN</td>
<td>Adacare</td>
<td>RMLP (Ours)</td>
</tr>
<tr>
<td>AUPRC</td>
<td>0.4532</td>
<td>0.3752</td>
<td>0.3094</td>
<td>0.4412</td>
<td><strong>0.6314</strong></td>
</tr>
<tr>
<td>AUROC</td>
<td>0.9013</td>
<td>0.8433</td>
<td>0.8579</td>
<td>0.8802</td>
<td><strong>0.9274</strong></td>
</tr>
<tr>
<td>LL</td>
<td>-0.3838</td>
<td>-0.2900</td>
<td>-0.1801</td>
<td>-0.1607</td>
<td><strong>-0.1270</strong></td>
</tr>
<tr>
<td>PPV-Small</td>
<td>0.6725</td>
<td>0.5825</td>
<td>0.5125</td>
<td>0.7250</td>
<td><strong>0.9300</strong></td>
</tr>
<tr>
<td>PPV-Medium</td>
<td>0.5590</td>
<td>0.5320</td>
<td>0.4150</td>
<td>0.5200</td>
<td><strong>0.7230</strong></td>
</tr>
<tr>
<td>PPV-Large</td>
<td>0.4907</td>
<td>0.4714</td>
<td>0.3736</td>
<td>0.4450</td>
<td><strong>0.6014</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DCD</th>
<th>Hemorrhagic Stroke Prediction</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Logistic Regression</td>
<td>LIME</td>
<td>RETAIN</td>
<td>Adacare</td>
<td>RMLP (Ours)</td>
</tr>
<tr>
<td>AUPRC</td>
<td>0.0830</td>
<td>0.0459</td>
<td>0.0271</td>
<td>0.0348</td>
<td><strong>0.1148</strong></td>
</tr>
<tr>
<td>AUROC</td>
<td><strong>0.8513</strong></td>
<td>0.7931</td>
<td>0.7265</td>
<td>0.7661</td>
<td>0.8213</td>
</tr>
<tr>
<td>LL</td>
<td>-0.419</td>
<td>-0.0827</td>
<td>-0.0613</td>
<td>-0.0621</td>
<td><strong>-0.0553</strong></td>
</tr>
<tr>
<td>PPV-Small</td>
<td>0.1800</td>
<td>0.0500</td>
<td>0.0600</td>
<td>0.0500</td>
<td><strong>0.2900</strong></td>
</tr>
<tr>
<td>PPV-Medium</td>
<td>0.1233</td>
<td>0.0700</td>
<td>0.0400</td>
<td>0.0600</td>
<td><strong>0.1733</strong></td>
</tr>
<tr>
<td>PPV-Large</td>
<td>0.1100</td>
<td>0.0725</td>
<td>0.0375</td>
<td>0.0475</td>
<td><strong>0.1450</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DCD</th>
<th>Transient Ischemic Attack Prediction</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Logistic Regression</td>
<td>LIME</td>
<td>RETAIN</td>
<td>Adacare</td>
<td>RMLP (Ours)</td>
</tr>
<tr>
<td>AUPRC</td>
<td>0.1839</td>
<td>0.1528</td>
<td>0.0773</td>
<td>0.0667</td>
<td><strong>0.2934</strong></td>
</tr>
<tr>
<td>AUROC</td>
<td><strong>0.8960</strong></td>
<td>0.8454</td>
<td>0.7609</td>
<td>0.7459</td>
<td>0.8831</td>
</tr>
<tr>
<td>LL</td>
<td>-0.3498</td>
<td>-0.0993</td>
<td>-0.1130</td>
<td>-0.0775</td>
<td><strong>-0.0611</strong></td>
</tr>
<tr>
<td>PPV-Small</td>
<td>0.3200</td>
<td>0.2900</td>
<td>0.2100</td>
<td>0.2300</td>
<td><strong>0.6800</strong></td>
</tr>
<tr>
<td>PPV-Medium</td>
<td>0.3000</td>
<td>0.2967</td>
<td>0.1833</td>
<td>0.1167</td>
<td><strong>0.4000</strong></td>
</tr>
<tr>
<td>PPV-Large</td>
<td>0.2800</td>
<td>0.2750</td>
<td>0.1700</td>
<td>0.0975</td>
<td><strong>0.3450</strong></td>
</tr>
</tbody>
</table>
From Table 2, we see that our RMLP model is the best performer on AUPRC (our main evaluation metric) for all the six tasks.

Comparing the performance of RMLP and logistic regression, we see that RMLP achieves better performances on nearly all metrics. Specifically, in terms of AUPRC, RMLP outperforms logistic regression by 7% on mortality prediction, 52% on decompensation prediction, 23% on eye complications, 39% on ischemic stroke prediction, 38% on hemorrhagic stroke prediction, and 60% on transient ischemic attack prediction. However, for the DCD prediction tasks (except eye complication), we notice that logistic regression performs better than RETAIN and Adacare in nearly all metrics except LL. This is in line with [58], which also observes that logistic regression frequently performs on par with sophisticated deep learning models on clinical prediction problems.\(^7\)

When comparing the performance of RMLP and LIME, we observe that RMLP achieves superior results across all metrics. Notably, in terms of AUPRC, RMLP surpasses LIME by 30% on mortality prediction, 153% on decompensation prediction, 42% on eye complications, 68% on ischemic stroke prediction, 150% on hemorrhagic stroke prediction, and 92% on transient ischemic attack prediction. These results demonstrate a significant advantage of RMLP over LIME, a post-hoc method that adjusts predictions from a trained MLP.

Comparing the baselines RETAIN and Adacare, we see that Adacare is the stronger performer that consistently achieves better results (except for transient ischemic attack). Comparing our RMLP model to (the strongest baseline) Adacare, we see that RMLP performs better than Adacare on all metrics for all tasks. Specifically, in terms of AUPRC, RMLP outperforms Adacare by 7% on mortality prediction, 7% on decompensation prediction, 12% on eye complications, 43% on ischemic stroke prediction, 230% on hemorrhagic stroke prediction, and 430% on transient ischemic attack prediction.

The better performance of RMLP against the state-of-the-art deep learning models, Adacare and RETAIN, is explained by (a) the ability of RMLP to model complex non-linear features in a sequential manner and (b) the ability of RMLP to learn WFSA patterns that model clinical events at different time-scales (e.g., this is helpful for modeling the blood pressure of an intensive-care patient which is taken hourly, and the medicinal prescription of a chronic-diseased patients which occurred monthly.)

We conduct paired t-tests to evaluate the statistical significance of the test log-likelihood results. The null hypothesis is that a baseline is at least as good as RMLP in terms of log-likelihood, and the alternative hypothesis is that RMLP is better. For the prediction tasks on eye complication, ischemic stroke, hemorrhagic stroke, and transient ischemic attack, we are able to reject the null hypothesis, and support the alternative hypothesis that RMLP is better than each baseline at the 95% and 99% confidence level. Similarly, we show that most of RMLP’s AUPRC and PPV results are statistically significantly better than those of the baselines at the 95% or 99% confidence level (specifically for decompensation prediction, eye complication prediction, ischemic stroke prediction, and transient ischemic attack prediction).

The remarkable performance of our RMLP model, in comparison to the baseline methods, strongly underscores its exceptional ability to precisely detect individuals with significant clinical risk factors. This accurate identification of high risk individuals not only allows for timely access to treatment but also promises to yield improved clinical outcomes and substantial reductions in healthcare costs.

To assess the computational load of our RMLP model in comparison to baseline models, we opted to compare with Adacare and RETAIN since they handle dynamic data, ensuring a fair and direct comparison. In this regard, we present both training and inference times of our RMLP model and

\(^7\)See Table 1 sequestered on the fourth last page of the paper’s supplementary material.
the baselines RETAIN and Adacare. These results are reported for the eye complications prediction task of the DCD dataset in Table 3.

Table 3. Training and Inference Times of Baselines Models and RMLP models on Eye Complication Task of the DCD dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Time Until Convergence</th>
<th>Inference Time per One Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETAIN</td>
<td>48 mins</td>
<td>0.027 secs</td>
</tr>
<tr>
<td>Adacare</td>
<td>1 hours, 22 minutes</td>
<td>0.025 secs</td>
</tr>
<tr>
<td>RMLP</td>
<td>2 hours, 13 minutes</td>
<td>0.030 secs</td>
</tr>
</tbody>
</table>

In Table 3, it is observed that while RMLP necessitates marginally more training time compared to the other two baselines, its inference time remains approximately the same.

5.4 Ablation Study

The implementation of our RMLP model differs from neural WFSA [62] in two key ways:

1. To ensure that WFSA leverages the most recent patient information for health risk predictions, we have restricted it from matching subsequences that do not conclude with the last segment of the input. This adjustment involves employing Equations 3, 4, and 5 to calculate $p_{\text{span}}(x)$ for a WFSA, instead of using Equations 3, 4, and 9.
2. We have also refined the transition weight function $T(\cdot)$, which was initially defined as a linear function of the input $x$ (e.g., $w_i \cdot x + b_i$) in Equation 6. The scalar output is further transformed by an encoding function $E$ such as the sigmoid function $\sigma$. To capture intricate non-linear interactions among input features, RMLP replaces the linear function with a Multi-Layer Perceptron (MLP), as illustrated in Equation 10. This adaptation allows for more flexible modeling of feature interactions in our health risk prediction tasks.

To conduct an ablation study, by solely implementing the first modification, we create the transition weight function $T(\cdot)$ using the sigmoid function as the encoding function $E$. This resulting model is denoted as Rational Logistic Regression or RLR in Table 4, as it can be demonstrated to be an extension of logistic regression specifically designed for sequential data, as shown in Proposition A.9. The selection of RLR is not random. As demonstrated in Section 5.3, logistic regression has exhibited promising outcomes in health risk prediction tasks, particularly on the DCD dataset. This substantiates our findings presented in Section A.1, which enable us to identify the suitable function for the specific problems we are addressing.

Note that we apply RMLP’s hyperparameters (detailed in Appendix A.2) where they are applicable for neural WFSA and RLR. As observed in Table 4, our RMLP model consistently outperforms all other models across all six tasks, as measured by the AUPRC, our primary evaluation metric.

The first modification, which introduces an inductive bias to leverage recent information, results in a substantial improvement in AUPRC. Comparing RLR and neural WFSA, this enhancement is noteworthy, with an increase of 8% for in-hospital mortality prediction, 127% for decompensation prediction, 63% for eye complications prediction, 64% for ischemic stroke prediction, 366% for hemorrhagic stroke prediction, and a remarkable 900% for transient ischemic attack prediction.

The second modification, which empowers the model’s capacity by replacing the transition weight function with a Multi-Layer Perceptron (MLP) to account for non-linear feature interactions, yields further improvements in AUPRC. This enhancement leads to an additional increase of 2% for in-hospital mortality prediction, 4% for decompensation prediction, 2% for eye complications...
Table 4. Empirical comparisons of RMLP against baseline systems. * and ** indicate statistical significance at the 95% and 99% level respectively.

<table>
<thead>
<tr>
<th>MIMIC-III</th>
<th>In-hospital Mortality Prediction</th>
<th>Decompensation Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Neural WFSA</td>
<td>RLR</td>
</tr>
<tr>
<td>AUPRC</td>
<td>0.4578</td>
<td>0.4933</td>
</tr>
<tr>
<td>AUROC</td>
<td>0.8470</td>
<td>0.8534</td>
</tr>
<tr>
<td>LL</td>
<td>-0.2673</td>
<td>-0.2619</td>
</tr>
<tr>
<td>PPV-Small</td>
<td>0.6400</td>
<td>0.7300</td>
</tr>
<tr>
<td>PPV-Medium</td>
<td>0.5333</td>
<td>0.5433</td>
</tr>
<tr>
<td>PPV-Large</td>
<td>0.4425</td>
<td><strong>0.4650</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DCD: Hemorrhagic Stroke Prediction</th>
<th>Transient Ischemic Attack Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Neural WFSA</td>
</tr>
<tr>
<td>AUPRC</td>
<td>0.0117</td>
</tr>
<tr>
<td>AUROC</td>
<td>0.5000</td>
</tr>
<tr>
<td>LL</td>
<td>-0.0639</td>
</tr>
<tr>
<td>PPV-Small</td>
<td>0.0100</td>
</tr>
<tr>
<td>PPV-Medium</td>
<td>0.0100</td>
</tr>
<tr>
<td>PPV-Large</td>
<td>0.0075</td>
</tr>
</tbody>
</table>

5.5 Interpretability

To find out the relative contributions among the $k$ patterns of an RMLP model to a prediction, we use a leave-one-out approach. Given the input sequence from a patient, RMLP produces $k$ scores (Equations 4 and 5), one for each pattern. These scores are fed to a final MLP (see Figure 5) to produce a (real-valued) probability prediction that a patient has an affliction. After zeroing-out the score of a pattern (effectively holding out its input to the final MLP), we note the change in the value produced by the final MLP. The value of this change reflects the contribution of the pattern to RMLP’s prediction. The $k$ patterns can thus be ranked in decreasing order of their associated changes, with the top-ranked pattern being the one that contributes the most to the prediction.

We investigate the interpretability of the WFSA patterns learned by our RMLP model at both the individual patient level and at the (sub)population level.

5.5.1 Individual Patient Level

In this section, we demonstrate the interpretability of patterns learned by our model for selected individuals with positive predictions, across the range of predictive tasks in our experiments. Moreover, we find that the patterns align well with the understanding of disease progression found in medical literature.

In-Hospital Mortality
We pick a patient $P_1$ who is predicted as a positive case from the validation set\(^8\) of the in-hospital mortality task.

The top pattern for patient $P_1$ is depicted in Figure 7(a). This pattern has 11 states (excluding states with $\epsilon$-transitions). This pattern is activated at time 14 (a time step corresponds to an hour). $P_1$ moves from state 0 to 1 because of a Glasgow Coma Scale (GCS) Motor Response score of 3. (Recall from Section 4.1 that each transition weight is calculated from an MLP. Therefore, we can identify the top input features for a transition (from the MLP) using SHapley Additive exPlanations (SHAP) values [44]. The top input features refer to those with the highest SHAP values, which collectively account for over 80% of the total summation of SHAP values. This is how we identify GCS Motor Response as an important feature for the first transition.) Next, $P_1$ moves from state 1 to 2 due to low systolic blood pressure (hypotension) at hour 15, defined as a systolic blood pressure lower than 100 mmHg [23]. $P_1$ then moves from state 2 to state 4 because of a GCS Motor Response score of 3. $P_1$ then sequentially progresses to state 10 because of a high heart rate (greater than 100 beats per minute [3]). $P_1$ loops for 25 time steps and stays at state 10 till time 48. In alignment with medical understanding, this pattern encapsulates the outcomes of a patient with likely traumatic brain injury [28, 42].

The pattern in Figure 7(a) illustrates how RMLP provides interpretability by threading together salient features into a coherent whole (previously mentioned in Section 2). This is in sharp contrast to the attention mechanism of a deep learning system that typically highlights important features across the entire timespan without clarifying how they relate to one another. For example, in

---

\(^8\)We do not use the test set to prevent us from inadvertently exposing our RMLP model to the test examples, and thereby compromising the veracity of the empirical results in Table 2.

ACM Trans. Manag. Inform. Syst., Vol. 0, No. 0, Article 0. Publication date: 2024.
Figure 7(a), the important features are Systolic Blood Pressure (at time 15), GCS Motor Response score (at time 14, 16, 17), and Heart Rate (at time 18 to 23). The attention mechanism merely highlights all these features as important without discerning whether all of them should form one regular sequence, or whether they should be broken up into subsets, each of which is pieced together to form a different sequence, such as:

1. GCS Motor Response$_{t=16}$ → GCS Motor Response$_{t=17}$
2. GCS Motor Response$_{t=14}$ → Systolic Blood Pressure$_{t=15}$

RMLP does not have such a problem. Indeed, RMLP provides a perspicuous view of how the features are linked together temporally in distinct sequences, and does not leave end-users wondering which combination of features go together in a pattern.

**Decompensation**

We repeat the analysis for a selected patient $P_2$ from the validation set of the decompensation prediction task. Figure 8(a) illustrates the most important pattern for $P_2$. $P_2$ moves from state 0 to 2 because of the persistent occurrences of both a high heart rate (higher than 100 beats per minute [3]) and low systolic blood pressure (lower than 100 mmHg [23]). This pattern is consistent with medical research [36], which suggests that the combination of low blood pressure and a high pulse rate may indicate inadequate oxygen supply to the body, possibly resulting in a severe condition such as shock. Moreover, as illustrated in Figure 8(b,c), the pattern emphasizes the end of the timeline, enabling it to capture the critical period preceding the patient’s death.

**Eye Complication**

Similar to the previous analyses conducted for the predictive tasks of in-hospital mortality and decompensation, we carry out an analysis for a selected patient $P_3$ from the validation set of the eye complication prediction task. The result is illustrated in Figure 9. A slow heart rate, defined as fewer than 60 beats per minute [5], is an important feature for the transitions from state 0 to 13. In addition, high systolic blood pressure (or hypertension, defined as systolic blood pressure above 140 mmHg [59]) is also a significant factor in the transition from state 10 to 13.

The identified pattern for $P_3$ aligns with medical research indicating that continuous high systolic blood pressure (or hypertension) can increase the risk of eye complications by 280% [24]. In addition
to utilizing the direct indication of hypertension, the pattern also takes advantage of an indirect indication of hypertension. Specifically, the pattern makes use of a slow heart rate, which is indicative of prolonged hypertension [5], to trigger state transitions.

**Ischemic Stroke**

We proceed with our analysis of a patient $P_4$ from the validation set of the ischemic stroke prediction task. We illustrate the result in Figure 10. The transition of $P_4$ from state 0 to 1 is triggered by a LDL cholesterol level of 3.66 mmol/l. Subsequently, the transition of $P_4$ from state 1 to 7 is attributed to consistent values of 941 pg/ml for N-terminal pro-brain natriuretic peptide (NT-proBNP). Note that we remove the states with epsilon transitions and self-loops to improve the readability of the pattern. According to the medical findings in Shweikialrefae et al. (2021; [64]), elevated levels of NT-proBNP (specifically, levels above 697 pg/ml as observed for the transitions from state 1 to 7) may indicate atrial fibrillation (AF) in patients. For patients with AF, LDL cholesterol levels above 1.5 mmol/l (as seen in the state transition from 0 to 1) are associated with an increased risk of ischemic stroke [64].

**Hemorrhagic Stroke**

Next, we analyze a selected patient $P_5$ from the validation set of the hemorrhagic stroke prediction task (Figure 11). Due to consistently high urea serum levels (defined as values greater than 9.2 mmol/l [48]), $P_5$ moves from state 0 to 8. Research [25] suggests that elevated levels of serum urea serve as an indicator of renal injury, which has been linked to an increased risk of hemorrhagic stroke. We observe in Figure 11(b) that the pattern takes advantage of the temporal information encoded in the urea serum input feature, particularly when its values are high, for state transitions.

**Transient Ischemic Attack**

Lastly, in Figure 12, we analyze patient $P_6$ from the validation set of the TIA prediction task. As his age progresses, $P_6$ moves from state 0 to 9. Also, platelet count, which is the number of platelets in the patient’s blood, plays a significant role in transition from state 0 to 2. The identified pattern is consistent with the established medical understanding of TIA, which occurs when a
blood vessel that supplies blood to the brain becomes blocked. Research indicates that as people age, their arteries naturally narrow, making age a potential risk factor for developing TIA [37]. Additionally, high platelet counts (more than 450,000/ml [18]) put patients at risk for blood clots obstructing blood vessels. It is worth noting that the pattern allows for transitions between states to occur at different intervals through the use of self-loops thereby accommodating two distinct age ranges in the age variable. Additionally, the pattern uses a smaller time scale for the platelet count observations, as indicated by the two consecutive timesteps for the transition from state 0 to 2. This is an illustration of how RMLP is able to model multi-time-scale events.

5.5.2 (Sub)population Level
To find the relative contributions of RMLP’s patterns at the (sub)population level, we repeat the leave-one-out approach for every patient, and find the average change in prediction value of each pattern across all patients. The pattern with the largest change in value is the one that contributes the most to RMLP’s predictions on average. Using this process, we identify the most important pattern on a (sub)population in the validation set of each predictive task.

**In-Hospital Mortality**

We first identify the pattern for a subpopulation that contains patients with positive predictions made by our RMLP model in the validation dataset. Then, following the approach used for individual-level interpretation in Section 5.5.1, we extract the top features with the largest SHAP values at the time of a state transition for each patient. The extracted features with largest SHAP values, which collectively account for 80% of the overall summation of SHAP values, are identified as important features associated with a state transition of this patient. To identify the important features at the subpopulation level, we next accumulate SHAP values for each important feature from all patients in the subpopulation. The important features at the subpopulation level are determined by selecting the largest features that collectively account for 80% of the total accumulated SHAP values of all the features. By identifying these important features at the subpopulation level for state transitions, we are able to calculate the average value for each feature.

In Figure 13, the binary variable indicating GCS score of 15 is found to be significant in triggering state transitions. GCS score ranges from 3 to 15, and a higher score indicates a healthier patient. It is evident from Figure 13 that the average value of the binary variable indicating GCS score 15 decreases as the pattern progresses. This decrease suggests a deterioration in patients’ consciousness, as evidenced by the decline in GCS scores. The deterioration in consciousness, together with abnormal vital signs, viz., hypotension (systolic blood pressure lower than 100 mmHg [23]), low respiratory rate (normal range is between 12 and 20 bpm [33]), and slightly low temperature (normal range is 36.1 – 37.8 °C [66]), is predictive of patient’s death.

**Decompensation**
We next conduct an analysis on decompensation’s subpopulation in the validation dataset whose constituent patients have positive predictions made by our RMLP model. We take the top features with the largest accumulated SHAP values, which together accounted for 60% of the total accumulated SHAP values. The result is shown in Figure 14. It involves 3 binary variables associated with GCS scores (mentioned in Section 5.5.1) that reflect the ability of a patient to move his eyes (GCS Eye Response), to move his body (GCS Motor Response), and to speak (GCS Verbal Response). Lower GCS scores are correlated with a higher risk of death [75]. In Figure 14, our pattern shows that it uses the binary variables indicating whether GCS Verbal Response Score or GCS Eye Response score is each equal to the lowest score of 1 (these variables are GCS Verbal==1 and GCS Eye==1 respectively). The other binary variable, on the other hand, indicates whether GCS Motor Response score is equal to the highest possible score of 6. For GCS Motor Response, we observe that the average scores of binary variables with high contributions toward state transitions are near zero, indicating that the most influential GCS Motor Response causing transitions have scores lower than 6. Furthermore, these scores decrease as state transitions progress, indicating that GCS Motor Response has a great influence on later state transitions and shows a higher proportion of scores below 6 compared to earlier state transitions. In contrast, the binary variables of GCS Verbal and Eye Response have relatively high average scores, suggesting that the influential GCS Verbal and Eye Response features causing transitions tend to have scores of 1. These scores increase as state transitions progress, implying that GCS Verbal Response or GCS Eye Response has a higher proportion of scores of 1 compared to those in earlier state transitions.

### Eye Complication

The number differs from other tasks, as our goal is to select the top features with the highest cumulative SHAP values, aiming for at least 50%. However, in other cases, a few features dominate, resulting in the cumulative SHAP values exceeding 80%. We would like to note that these accumulated SHAP value for top features involves a trade-off between simplicity and coverage. While a lower summation may also lead to a reduced coverage of features that affects state transitions, it would make the chosen features simpler to comprehend.
As with our previous analyses, we examine a subpopulation of patients predicted by the RMLP model to have eye complications in the validation dataset. The findings depicted in Figure 15 indicate that there is a correlation between consistent hypercholesterolemia and the development of eye complications. This is due to the fact that hypercholesterolemia poses a risk factor for chronic ischemia in the retina, which in turn underpins the pathophysiology in diabetic retinopathy and retinal vascular occlusive disease. These correlations are substantiated by medical research [43, 70].

**Ischemic Stroke**

The next analysis focuses on the male subpopulation, for whom the RMLP model makes positive predictions within the validation dataset. The resulting pattern, as shown in Figure 16(a), aligns with medical knowledge that hypertension is a significant risk factor for the development of ischemic stroke [8].

**Hemorrhagic Stroke**

We conduct an examination of the male subpopulation who received a positive prediction from our RMLP model. As shown in Figure 16(b), we notice that the pattern incorporates a binary variable to ascertain whether a patient is Chinese when transitioning from state 0 to 2. This observation is consistent with existing medical literature [74], which states that Chinese men in Singapore have the highest prevalence rate of stroke when compared to other ethnic groups.

**Transient Ischemic Attack**

Finally, we conduct an analysis for transient ischemic attack. Our focus is on a subpopulation of women over 50 years old who are identified by our RMLP model with positive predictions. The results are shown in Figure 16(c). The pattern in the figure depicts high levels of urea serum (normally 3.5-7.2 mmol/l [48]) followed by advanced age. According to a study [19], transient ischemic attack (TIA) is associated with an increased risk of urinary tract infections (UTIs), which could consequently lead to elevated urea levels. Therefore, the detection of high urea levels by the pattern suggests the likely presence of a UTI [55], which in turn is indicative of TIA. Additionally, as noted in Section 5.5.1 and supported by [37], advanced age constitutes another risk factor of TIA. The state transitions in the pattern are well-aligned with these medical findings.
It is important to emphasize that the validation of RMLP’s interpretation primarily relies on medical literature. While the medical literature aligns with the illustrated patient cases, there are limitations associated with this approach. This is because the patients featured in our illustrations may have unique circumstances, such as varying medical conditions or treatments, which can introduce inaccuracies when extrapolating conclusions from diverse patient cases.

To establish a more rigorous validation, one could involve clinicians to determine whether the algorithm’s inferred patterns align with or assist in their decision-making processes for individual patients. Clinician input can provide valuable insights into the clinical relevance and applicability of the identified patterns.

6 Conclusion and Future Work

In this paper, we present RMLP, a novel model that predicts clinical events from EHR data, and learns interpretable patterns as weighted finite state automata (WFSAs). We contribute to the healthcare domain via RMLP as a designed artifact. Our RMLP model not only effectively identifies patients at high clinical risk but also provides interpretable predictions. Furthermore, we prove that RMLP is a generalization of multi-layer perceptron to longitudinal time-series data. RMLP chains multiple MLPs together, and stacks these chains on top of each other. Additionally, we show how it encompasses neural WFSAs as a specific instance. Importantly, this generalization can be expanded to encompass other static models, as demonstrated in Appendix A.8. This provides a fundamental principle for extending a static model into a dynamic one through the utilization of WFSA.

The self-loops and \( \epsilon \)-transitions in the WFSAs underlying RMLP allow it to flexibly learn patterns across different time-scales. Experimental comparisons of RMLP against logistic regression, LIME, and two state-of-the-art deep learning models on six real-world tasks demonstrate the promise of our approach. Empirical analysis of RMLP’s learned patterns shows the meaningfulness of the knowledge that is captured by the patterns. In terms of interpretability, RMLP surpasses existing methods by chaining together features into coherent, distinct sequential patterns. This is in contrast to extant methods such as attention mechanisms of deep learning systems that merely highlight important features across an entire timespan without explicating the possibly multifarious sequential structures interweaving among them.

As future work, we want to use evolutionary algorithms to search for the number of WFSAs and their lengths, rather than specifying these as hyperparameters. We also want to move beyond WFSAs, and incorporate more powerful automata higher in the Chomsky hierarchy [17] into RMLP. Lastly, to further validate the interpretability of our results, we aim to seek clinical verification. We plan to involve clinicians in assessing whether the specific patterns identified by our RMLP model align with their existing knowledge or even assist them in making medical decisions.

Acknowledgments

We would like to express our sincere gratitude to the anonymous reviewers for their insightful feedback and constructive comments. These contributions greatly improved the quality of this manuscript.

This research is partly supported by Singapore Ministry of Health’s National Innovation Challenge Grant (MOH/NIC/CDM1/2018) and Singapore Ministry of Education’s AcRF Tier 1 Grant (R-253-000-146-133) to Stanley Kok. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors, and do not reflect the views of the funding agencies.
References


APPENDIX

A.1 Generalization of Static Models to Sequential Data

A.1.1 Proof of Theorem A.8

In this section, we aim to demonstrate that the WFSA is capable of generalizing a static non-negative model to sequential data. We present our main result in Theorem A.8. Before delving into the proof, let us first define what we mean by static data, sequential data, static non-negative model, and its generalization to sequential data. Additionally, we will introduce some lemmas that are essential in establishing Theorem A.8.

Definition A.1 (Static Data). Consider a real-valued vector $\tilde{x}$ of size $J$, and $\tilde{x}_j$ denotes the $j^{th}$ element of $\tilde{x}$. If $\tilde{x}$ represents data for a single instance where $\forall j \in \{1, \ldots, J\}$, an element $\tilde{x}_j$ represents the value of the $j^{th}$ feature out of $J$ total features, then $\tilde{x}$ is a static data.

In broad sense, static data can be defined as a collection of values that do not vary over time, and that can be arranged in a real-valued vector where each element corresponds to a particular feature.

Definition A.2 (Sequential Data). Let $x$ be a sequence of $n$ real-valued vectors, each of $J$ dimensions. We denote the $t^{th}$ vector in the sequence by $x_t$, and use the notation $x_{t,j}$ to represent the entry at the $j^{th}$ position of the $t^{th}$ vector. If $x$ represents data for a single instance where $\forall t \in \{1, \ldots, n\}$ and $\forall j \in \{1, \ldots, J\}$, $x_{t,j}$ represents the value of the $j^{th}$ feature at timestep $t^{th}$, then $x$ is a sequential data.

Generally, the sequential data means that the data can change over time, and the changes are captured along a sequence of vectors. Each vector in the sequence represents the values of the features at a particular time point, and the sequence captures the evolution of the feature values over time. An example of sequential data, as defined in Definition A.2, is a sequential input $x = (x_1, \ldots, x_n)$ in which $x$ represents a series of patient observations. Each element, $x_t$, in the sequence is a vector containing two binary variables $x_t = (\text{Vomit&Fever}_t, \text{Diarrhea}_t)$. The variable $\text{Vomit&Fever}_t$ indicates whether the patient is experiencing both vomiting and fever at time step $t$, while Diarrhea, specifies if the patient has diarrhea at the same time step.

Definition A.3 (Static Non-Negative Model). A function $\lambda$ is a static non-negative model if it is a function that produces non-negative real numbers as output and can only take static data in Definition A.1 as input.

The class of models defined in Definition A.3 includes well-established functions commonly employed with static data, such as logistic regression, quadratic function, or multi-layer perceptrons, where specific activation functions in the output layer, such as ReLU or sigmoid, ensure non-negative outputs. By having static data such as $\tilde{x} = (\text{Vomit&Fever}, \text{Diarrhea})$, where $\text{Vomit&Fever}$ and $\text{Diarrhea}$ represent whether a patient experienced fever accompanied by vomiting, or diarrhea on any day during their hospital stay, one can employ this type of model to predict the probability of an individual experiencing gastroenteritis, one of its causes being rotavirus [34], which clinically presents with vomiting and fever followed by diarrhea. Two logistic regression models can be employed for this purpose. One model evaluates the risk associated with having fever accompanied by vomiting, denoted as $\lambda(\text{Vomit&Fever}; G_1)$, while the other measures the risk associated with diarrhea, denoted as $\lambda(\text{Diarrhea}; G_2)$. Note that $G_1$ and $G_2$ are parameters specific to the logistic regression models, indicating that they represent two distinct logistic regression models. The overall risk can be computed by multiplying these individual risks: $\lambda(\text{Vomit&Fever}; G_1) \cdot \lambda(\text{Diarrhea}; G_2)$.
We will now illustrate how these models, traditionally designed for static data, can be extended to sequential data. Consider a sequential input \( x = (x_1, x_2, \ldots, x_n) \) containing information for each patient. As previously stated, let \( x_i = (\text{Vomit}&\text{Fever}_i, \text{Diarrhea}_i) \). Reiterating, \( \text{Vomit}&\text{Fever}_i \) and \( \text{Diarrhea}_i \) are binary variables indicating whether a patient is currently experiencing a combination of fever and vomiting or has diarrhea at timestep \( i \).

To utilize a static model \( \lambda \) for predicting the probability of an individual experiencing gastroenteritis, as previously described, we choose a subsequence that captures essential information. For instance, we may opt for the most recent data from the last two timesteps, i.e., \( (x_5, x_6) \), given that gastroenteritis could clinically manifest with fever accompanied by vomiting in the first timestep, followed by diarrhea in the next timestep.

The static model \( \lambda \) is subsequently applied to each element of this subsequence, namely, \( \lambda(x_5;G_1) \cdot \lambda(x_6;G_2) \). Here, \( \lambda(x_5;G_1) \) assesses the risk of gastroenteritis related to fever and vomiting, while \( \lambda(x_6;G_2) \) considers the risk associated with diarrhea. This extension is formally defined and referred to as the GSMSD(\( \lambda \)) Function in Definition A.4.

**Definition A.4 (GSMSD(\( \lambda \)) Function).** A function \( f \) is a generalization of a static non-negative model, \( \lambda(\cdot) \), to sequential data (or a GSMSD(\( \lambda \)) function in short) if it has the following form:

\[
f(x) = C \lambda(x_1;G_1) \lambda(x_2;G_2) \cdots \lambda(x_n;G_N),
\]

where sequential data (Definition A.2) \( x = (x_1, \ldots, x_n) \) is a sequence of inputs; \( x_i \) is a real-valued vector \( i \in \{1, \ldots, n\} \); \( \lambda(y;\theta) \) is the output of a static non-negative model \( \lambda \) (as in Definition A.3) with parameters \( \theta \) of input \( y; (i_1, \ldots, i_N) \) is a subsequence of \( (1, \ldots, n) \); and \( C \) is a non-negative real number.

Having clarified the concept of extending a static model to sequential data as defined in Definition A.4, our next step is to demonstrate that the WFSA has the ability to generalize a static non-negative model to sequential data. This serves as our main result, outlined in Theorem A.8, and functions as a guiding principle for extending a static model to sequential data. However, before delving into the proof of this theorem, we will present and establish certain lemmas essential for proving Theorem A.8.

**Lemma A.5.** If \( C' \) is a non-negative real number, and \( f(x) \) is a GSMSD(\( \lambda \)) function, then \( C'f(x) \) is also a GSMSD(\( \lambda \)) function.

**Proof.** Let \( f(x) = C \lambda(x_1;G_1) \lambda(x_2;G_2) \cdots \lambda(x_n;G_N) \).

\[
C'f(x) = C'' \lambda(x_1;G_1) \lambda(x_2;G_2) \cdots \lambda(x_n;G_N)
\]

(\( C'' = C'C \) is a non-negative real number).

**Lemma A.6.** A non-negative real number \( C \) is a (degenerate) GSMSD(\( \lambda \)) function.

**Proof.** In Equation 14 from Definition A.4, let \( (i_1, \ldots, i_N) \) be an empty subsequence.

Thus, \( C \lambda(x_1;G_1) \lambda(x_2;G_2) \cdots \lambda(x_N;G_N) = C \).

**Lemma A.7.** If \( f(x) \) is a GSMSD(\( \lambda \)) function (where \( x = (x_1, \ldots, x_N) \)), and \( g(x_{N+1}) = \lambda(x_{N+1}; w) \) is an output from a static non-negative model \( \lambda \) of input \( x_{N+1} \), then \( f(x)g(x_{N+1}) \) is a GSMSD(\( \lambda \)) function.

**Proof.** Since \( f(x) \) is a GSMSD(\( \lambda \)) function, it can be expressed as \( f(x) = C \lambda(x_1;G_1) \lambda(x_2;G_2) \cdots \lambda(x_N;G_N) \).

ACM Trans. Manag. Inform. Syst., Vol. 0, No. 0, Article 0. Publication date: 2024.
Thus, \( f(x)g(x_{N+1}) = C\lambda(x_1; G_1)\lambda(x_2; G_2)\cdots\lambda(x_{i_N}; G_N)\lambda(x_{N+1}; w) \). Since \((i_1, \ldots, i_N, N + 1)\) is a subsequence of \((1, \ldots, N + 1)\), \( f(x)g(x_{N+1}) \) is a GSMSD(\(\lambda\)) function by Definition A.4.

We are now prepared to present our main result of this section in Theorem A.8, which demonstrates that the WFSA has the capability to generalize a static non-negative model \(\lambda\) to sequential data by defining the transition weight function \(T(\cdot)\) as

\[
[T(x)]_{i,j} = \begin{cases} 
\lambda(x; u_i) & \text{if } j = i \text{ (self-loop)}, \\
\lambda(x; w_i) & \text{if } j = i + 1, \\
0 & \text{otherwise},
\end{cases}
\]

(15)

\[
[T(\epsilon)]_{i,j} = \begin{cases} 
\gamma(c_i) & \text{if } j = i + 1, \\
0 & \text{otherwise},
\end{cases}
\]

(16)

where \(\lambda(x; \theta)\) is an output from a static non-negative model of input \(x\) with parameters \(\theta\). \(\gamma(c)\) is a non-negative function of \(c\). Another relevant aspect to consider when extending a static model to sequential data is the subsequence of interest, as outlined in Definition A.4. In our approach, we selected the max-product semiring to compute a score \(p_{\text{span}}\) for WFSA with input \(x\) using Equations 3, 4, and 5. As detailed in Section 3.3, the utilization of Equations 3, 4, and 5 enables us to take into account all subsequences that end with the last input of the sequence. Then, we pick the subsequence (along with its associated path) that produces the highest score.

**Theorem A.8.** The function \(p_{\text{span}}(x)\) as defined by the following Equations 3, 4, 5, 15, and 16 using the max-product semiring \((\mathbb{R}_{\geq 0}, \max, 0, x, 1)\) is a GSMSD(\(\lambda\)) function.

\[
h_0 = \pi \ T(\epsilon)^* = \pi \ \max(I, T(\epsilon))
\]

(3)

\[
h_{t+1} = \max \ (h_t \ T(x_{t+1})) \ T(\epsilon)^*, \ h_0)
\]

\[
= \max (h_t \ T(x_{t+1})) \max(I, T(\epsilon)), \ h_0
\]

(4)

\[
p_{\text{span}}(x) = h_n \eta
\]

(5)

In the above equations, we have approximated \(\pi \ T(\epsilon)^*\) as \(\pi \ \max(I, T(\epsilon))\) (as mentioned in Section 4.1), and \(\pi = [\ 1 \ 0 \ \cdots \ 0 \ ]_{(1 \times D)}\) is an initial weight vector, \(\eta = [\ 0 \ \cdots \ 1 \ ]^T \in \mathbb{R}^{D \times 1}\) is a final weight vector, \(D\) is the number of states, and the transition weight function \(T(\cdot)\) is

\[
[T(x)]_{i,j} = \begin{cases} 
\lambda(x; u_i) & \text{if } j = i \text{ (self-loop)}, \\
\lambda(x; w_i) & \text{if } j = i + 1, \\
0 & \text{otherwise},
\end{cases}
\]

(15)

\[
[T(\epsilon)]_{i,j} = \begin{cases} 
\gamma(c_i) & \text{if } j = i + 1, \\
0 & \text{otherwise},
\end{cases}
\]

(16)

where \(\lambda(x; \theta)\) is an output from a static non-negative model of input \(x\) with parameters \(\theta\). \(\gamma(c)\) is a non-negative function of \(c\).

**Proof.** Observe that \(h_t = [h_{t,0}, h_{t,1}, \ldots, h_{t,D-1}]\) is a \(D\)-dimensional row vector for \(t \in \{1, \ldots, n\}\). We shall show with mathematical induction on \(t\) that each element \(h_{t,i} (i \in \{0, \ldots, D - 1\})\) in the row vector is a GSMSD(\(\lambda\)) function. Note that because we are using the max-product semiring, the sum operator in matrix/vector multiplication is replaced by the max operator.
As base case, we substitute the transition weight function $T(\cdot)$ defined in Equation 16 into Equation 3, and explicitly evaluate $h_0$ as

$$h_0 = \pi \max(I, T(\epsilon)) = \begin{bmatrix} 1 & 0 & \cdots & 0 & 1 \gamma(c_0) & 0 & 0 & \cdots & 0 \end{bmatrix}.$$  

By inspection, we see that each element $h_{0,t}$ of $h_0$ is a non-negative real number, and thus is a GSMD$\hat{(}\lambda)$ function by Lemma A.6.

Before showing the inductive step, we first derive the explicit form of $h_{t+1}$ by substituting the transition weight functions $T(\cdot)$ defined in Equations 15 and 16 into Equation 4 (reshown here for convenience):

$$h_{t+1} = \max( (h_t T(x_{t+1})) \max(I, T(\epsilon)), h_0 ).$$

The term $h_t T(x_{t+1})$ on the right-hand side is evaluated as

$$h_t T(x_{t+1}) = \begin{bmatrix} h_{t,0} & \cdots & h_{t,D-1} \end{bmatrix} \begin{bmatrix} \lambda(x_{t+1}; u_0) & \lambda(x_{t+1}; w_0) & 0 & 0 \\ 0 & \lambda(x_{t+1}; w_{D-2}) & \lambda(x_{t+1}; u_{D-1}) \\ 0 & 0 & \lambda(x_{t+1}; u_{D-1}) \\ \end{bmatrix}$$

Using Equation 18, we next evaluate the first argument of the outer $\max(\cdot, \cdot)$ in Equation 4 as

$$h_t T(x_{t+1}) \max(I, T(\epsilon)) = h_t T(x_{t+1}) \begin{bmatrix} 1 & \gamma(c_0) & 0 & \cdots & 0 & 1 \gamma(c_0) & 0 & 0 & \cdots & 0 \end{bmatrix}.$$  

Using the results from Equations 17 and 19, we derive the explicit form of $h_{t+1}$ as

$$h_{t+1} = \max( (h_t T(x_{t+1})) \max(I, T(\epsilon)), h_0 )$$

Observe that the elements in $h_{t+1}$ take one of three forms.

1. The $0^\text{th}$ element is $h_{t+1,0} = \max( h_{t,0} \lambda(x_{t+1}; u_0), 1 ).$
2. The $1^\text{st}$ element is $h_{t+1,1} = \max( \gamma(c_0) h_{t,0} \lambda(x_{t+1}; u_0), h_{t,1} \lambda(x_{t+1}; u_1), h_{t,0} \lambda(x_{t+1}; w_0), \gamma(c_0) ).$
3. All other elements \( h_{t+1,i} (i \in \{2, \ldots, D - 1\}) \) takes the form
\[
h_{t+1,i} = \max( y(c_{t})h_{t,i-1} \lambda(x_{t+1}; w_{i-1}), y(c_{t})h_{t,i-2} \lambda(x_{t+1}; w_{i-2}), h_{t,i} \lambda(x_{t+1}; u_{i}), h_{t,i-1} \lambda(x_{t+1}; w_{i-1})).
\]

We shall next apply the inductive step to each of the three cases in turn. For each case, we assume that every element \( h_{t,i} \) in \( h_{t} \) is a GSMSD(\( \lambda \)) function, and show that consequently every element \( h_{t+1,i} \) in \( h_{t+1} \) is also a GSMSD(\( \lambda \)) function.

(1) In \( h_{t+1,0} = \max(h_{t,0} \lambda(x_{t+1}; u_{0}), 1) \), we see that \( h_{t,0} \lambda(x_{t+1}; u_{0}) \) is a GSMSD(\( \lambda \)) function by the inductive hypothesis and Lemma A.7, and 1 is a GSMSD(\( \lambda \)) function by Lemma A.6. Since both arguments in the \( \max(\cdot, \cdot) \) function are GSMSD(\( \lambda \)) functions, \( h_{t+1,0} \) is a GSMSD(\( \lambda \)) function.

(2) For \( h_{t+1,1} = \max( y(c_{0})h_{t,0} \lambda(x_{t+1}; u_{0}), h_{t,1} \lambda(x_{t+1}; u_{1}), h_{t,0} \lambda(x_{t+1}; w_{0}), y(c_{0}) ) \), we shall consider each argument in the max function in turn.

(a) In the first argument \( y(c_{0})h_{t,0} \lambda(x_{t+1}; u_{0}) \), we see that \( y(c_{0}) \) is a non-negative real number. Since \( h_{t,0} \lambda(x_{t+1}; u_{0}) \) is a GSMSD(\( \lambda \)) function by the inductive hypothesis, \( y(c_{0})h_{t,0} \lambda(x_{t+1}; u_{0}) \) is also a GSMSD(\( \lambda \)) function by Lemma A.7. Hence, \( y(c_{0})h_{t,0} \lambda(x_{t+1}; u_{0}) \) is also a GSMSD(\( \lambda \)) function by Lemma A.6.

(b) For the second argument \( h_{t,1} \lambda(x_{t+1}; u_{1}) \) and the third argument \( h_{t,0} \lambda(x_{t+1}; w_{0}) \), we use Lemma A.7 to conclude that they are both GSMSD(\( \lambda \)) functions.

(c) The fourth argument \( y(c_{0}) \) is a non-negative real number and is thus a GSMSD(\( \lambda \)) function by Lemma A.6.

Since all arguments of the max function are GSMSD(\( \lambda \)) functions, \( h_{t+1,1} \) is also a GSMSD(\( \lambda \)) function.

(3) For every remaining element \( h_{t+1,i} \) below (where \( i \in \{2, \ldots, D - 1\} \)), we consider each of the four arguments in its max function in turn.
\[
h_{t+1,i} = \max( y(c_{t})h_{t,i-1} \lambda(x_{t+1}; w_{i-1}), y(c_{t})h_{t,i-2} \lambda(x_{t+1}; w_{i-2}), h_{t,i} \lambda(x_{t+1}; u_{i}), h_{t,i-1} \lambda(x_{t+1}; w_{i-1})).
\]

(a) In the first argument \( y(c_{t})h_{t,i-1} \lambda(x_{t+1}; w_{i-1}) \) and second argument \( y(c_{t})h_{t,i-2} \lambda(x_{t+1}; w_{i-2}) \), observe that \( y(c_{t}) \) is a non-negative real number. We thus apply Lemmas A.5 to show that both \( y(c_{t})h_{t,i-2} \) and \( y(c_{t})h_{t,i-2} \) are GSMSD(\( \lambda \)) functions. Next we apply Lemma A.7 to conclude that both first and second arguments are GSMSD(\( \lambda \)) functions.

(b) In the third argument \( h_{t,i} \lambda(x_{t+1}; u_{i}) \) and fourth argument \( h_{t,i-1} \lambda(x_{t+1}; w_{i-1}) \), both \( h_{t,i} \) and \( h_{t,i-1} \) are GSMSD(\( \lambda \)) functions by the inductive hypothesis. By using Lemma A.7, we conclude that both third and fourth arguments are GSMSD(\( \lambda \)) functions.

Since all four arguments of the max function are GSMSD(\( \lambda \)) functions, \( h_{t+1, i} \) is also a GSMSD(\( \lambda \)) function.

From Equation 5, we see that \( p_{\text{span}}(x) = h_{n} \eta = \max_{i=0}^{D-1} h_{n,i} \eta_{i} \), where \( \eta \) is \( D \)-dimensional column vector of real values, and \( h_{n} \) is \( D \)-dimensional row vector of GSMSD(\( \lambda \)) functions. Since \( h_{n,i} \) is a GSMSD(\( \lambda \)) function, \( h_{n,i} \eta_{i} \) is a GSMSD(\( \lambda \)) function by Lemma A.5. Consequently, \( p_{\text{span}}(x) \) is a GSMSD(\( \lambda \)) function.

A.1.2 Proof of Proposition A.9

This section aims to provide a mathematical proof demonstrating the neural WFSA [62] can serve as a generalization of logistic regression to sequential data. Given the result in Theorem A.8, such a result is anticipated, since it utilizes logistic regression models as its transition weight functions.
Specifically, we show that by using the sigmoid function, \( \sigma(\cdot) \), as the encoding function under the max-product semiring \((\mathbb{R}_{\geq 0}, \max, 0, 1)\), the neural WFSA can achieve this generalization.

**Proposition A.9.** The function \( p_{\text{span}}(x) \) as defined by the following Equations 3, 4, 5, 21, and 22 using the max-product semiring \((\mathbb{R}_{\geq 0}, \max, 0, 1)\) is a GSMSD(Logistic Regression) function. In particular,

\[
\begin{align*}
  h_0 &= \pi T(e)^* = \pi \max(I, T(e)) \\
  h_{t+1} &= \max( (h_t T(x_{t+1})) T(e)^*, h_0 ) \\
  \cdots &= \max( (h_t T(x_{t+1})) \max(I, T(e)), h_0 )
\end{align*}
\]

where we have approximated \( \pi T(e)^* \) as \( \pi \max(I, T(e)) \) (as mentioned in Section 4.1), and \( \pi = [1 \ 0 \ \cdots \ 0] \in \mathbb{R}^{1 \times D} \) is an initial weight vector, \( \eta = [0 \ 0 \ \cdots \ 1]^T \in \mathbb{R}^{D \times 1} \) is a final weight vector, \( D \) is the number of states, and the transition weight function \( T(\cdot) \) is

\[
[T(x)]_{i,j} = \begin{cases} 
  \sigma(A_i \cdot x + a_i) & \text{if } j = i \ (\text{self-loop}), \\
  \sigma(B_i \cdot x + b_i) & \text{if } j = i + 1, \\
  0 & \text{otherwise},
\end{cases}
\]

\[
[T(e)]_{i,j} = \begin{cases} 
  \sigma(c_i) & \text{if } j = i + 1, \\
  0 & \text{otherwise},
\end{cases}
\]

where \( \sigma(y) := \frac{1}{1+e^{-y}} \).

**Proof.** The proof is a direct application of Theorem A.8 in which we define the transition weight function \( T(\cdot) \) in Equation 15 using the logistic regression since its output is non-negative, i.e.,

\[
[T(x)]_{i,j} = \begin{cases} 
  \lambda(x; u_i) := \sigma(A_i \cdot x + a_i) & \text{if } j = i \ (\text{self-loop}), \\
  \lambda(x; w_i) := \sigma(B_i \cdot x + b_i) & \text{if } j = i + 1, \\
  0 & \text{otherwise}.
\end{cases}
\]

Likewise, we define the transition weight function \( T(\cdot) \) in Equation 16 using the sigmoid function, i.e.,

\[
[T(e)]_{i,j} = \begin{cases} 
  \gamma(c_i) := \sigma(c_i) & \text{if } j = i + 1, \\
  0 & \text{otherwise}.
\end{cases}
\]

\[ \square \]

**A.1.3 Proof of Proposition A.10**

In this section, our goal is to present a mathematical proof that establishes RMLP as a generalization of MLP to sequential data. Similar to Proposition A.9, which establishes that the neural WFSA generalizes logistic regression to sequential data, RMLP employs MLP as its transition weight functions instead of the logistic regression model. Consequently, it is anticipated that RMLP serves as a generalization of MLP to sequential data.

**Proposition A.10.** The function \( p_{\text{span}}'(x) \) as defined by the following Equations 3, 4, 5, 22, and 23 using the max-product semiring \((\mathbb{R}_{\geq 0}, \max, 0, x, 1)\) is a GSMSD(Multi-Layer Perceptron) function. In
particular,
\[ h_0 = \max \{ T(e)^* \} = \max (I, T(e)) \]  
\[ h_{t+1} = \max (\{ h_t T(x_t) \} T(e)^*, h_0) \]
\[ = \max (\{ h_t T(x_{t+1}) \} \max (I, T(e)), h_0) \]
\[ p_{\text{span}}'(x) = h_{n-\eta} \]
where we have approximated \( \pi T(e)^* \) as \( \pi \max (I, T(e)) \) (as mentioned in Section 4.1), and \( \pi = [0 \cdots 0 \ 1]_{(1 \times D)} \) is an initial weight vector, \( \eta = [0 \cdots 0 \ 1] \in \mathbb{R}^{D \times 1} \) is a final weight vector, \( D \) is the number of states, and the transition weight function \( T(\cdot) \) is defined by

\[ [T(x)]_{i,j} = \begin{cases} 
\text{MLP}(x; u_i) & \text{if } j = i \text{ (self-loop)}, \\
\text{MLP}(x; w_i) & \text{if } j = i+1, \\
0 & \text{otherwise},
\end{cases} \]

\[ [T(e)]_{i,j} = \begin{cases} 
\sigma(c_i) & \text{if } j = i+1, \\
0 & \text{otherwise},
\end{cases} \]

where MLP(\( x; \theta \)) is the output of MLP given input \( x \) and parameters \( \theta \) including every layer’s weights and biases and the sigmoid function (or \( \sigma(\cdot) \)) is used for the non-linear activation in the last layer.

PROOF. The proof is a direct application of Theorem A.8 where we define the transition weight function \( T(\cdot) \) in Equation 18 using the MLP with the sigmoid function in the last layer since its output is non-negative, i.e.,

\[ [T(x)]_{i,j} = \begin{cases} 
\lambda(x; u_i) := \text{MLP}(x; u_i) & \text{if } j = i \text{ (self-loop)}, \\
\lambda(x; w_i) := \text{MLP}(x; w_i) & \text{if } j = i+1, \\
0 & \text{otherwise}.
\end{cases} \]

Likewise, we define the transition weight function \( T(\cdot) \) in Equation 16 using the sigmoid function, i.e.,

\[ [T(e)]_{i,j} = \begin{cases} 
y(c_i) := \sigma(c_i) & \text{if } j = i+1, \\
0 & \text{otherwise}.
\end{cases} \]

\( \square \)

A.1.4 Proof of Proposition A.11
The aim of this section is to provide a proof that RMLP encompasses neural WFSA as its specific instance.

PROPOSITION A.11. Function \( p_{\text{span}}(x) \) is a special case of the function \( p_{\text{span}}'(x) \) where \( p_{\text{span}}(x) \) is defined using Equations 3, 4, 5, 21, 22, and the max-product semiring \( (\mathbb{R}_{\geq 0}, \max, 0, x, 1) \) and shown to be a GSMSD(Logistic Regression) function by Proposition A.9. While the function \( p_{\text{span}}'(x) \) is formulated using Equations 3, 4, 5, 23, 22, and the max-product semiring \( (\mathbb{R}_{\geq 0}, \max, 0, x, 1) \), and Proposition A.10 has verified it to be a GSMSD(Multi-Layer Perceptron) function.

PROOF. Instead of defining the transition weight function \( T(\cdot) \) as Equation 21, \( p_{\text{span}}'(x) \) is formulated by using Equation 23 or

\[ [T(x)]_{i,j} = \begin{cases} 
\text{MLP}(x; u_i) & \text{if } j = i \text{ (self-loop)}, \\
\text{MLP}(x; w_i) & \text{if } j = i+1, \\
0 & \text{otherwise}.
\end{cases} \]
By using a single-layered neural network with a sigmoid non-linear activation function as $MLP$ in Equation 23, the transition weight function $T(\cdot)$ becomes

$$
[T(x)]_{i,j} = \begin{cases} 
MLP(x; u_i) := \sigma(A_i \cdot x + a_i) & \text{if } j = i \text{ (self-loop)}, \\
MLP(x; w_i) := \sigma(B_i \cdot x + b_i) & \text{if } j = i + 1, \\
0 & \text{otherwise},
\end{cases}
$$

(24)

where $u_i := (A_i, a_i)$, and $w_i := (B_i, b_i)$. The derived transition weight function in Equation 24 is identical to the function employed by Equation 21 to formulate $p_{span}(x)$. As a result, $p_{span}'(x)$ subsumes $p_{span}(x)$ as its special case.

### A.2 RMLP Model Hyperparameters

In this section, we provide a summary of the hyperparameters of our RMLP model that were tuned for different datasets using a validation set in Table 5.

<table>
<thead>
<tr>
<th>Task</th>
<th>WFSA</th>
<th>MLP (Transition Weight Function)</th>
<th>final MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-hospital Mortality (MIMIC-III)</td>
<td>$7 \times 3$-state, $4 \times 6$-state, $7 \times 9$-state, $7 \times 12$-state, $2 \times 15$-state, $7 \times 18$-state</td>
<td>2-layer (30 neurons)</td>
<td>1-layer</td>
</tr>
<tr>
<td>Decompensation (MIMIC-III)</td>
<td>$10 \times 3$-state, $10 \times 4$-state, $10 \times 5$-state</td>
<td>3-layer (30, 10 neurons)</td>
<td>1-layer</td>
</tr>
<tr>
<td>Eye Complications (DCD)</td>
<td>$7 \times 3$-state, $5 \times 6$-state, $4 \times 12$-state, $3 \times 15$-state</td>
<td>3-layer (175, 115 neurons)</td>
<td>1-layer</td>
</tr>
<tr>
<td>Ischemic Stroke (DCD)</td>
<td>$6 \times 2$-state, $6 \times 5$-state, $7 \times 10$-state, $7 \times 20$-state</td>
<td>2-layer (10 neurons)</td>
<td>1-layer</td>
</tr>
<tr>
<td>Hemorrhagic Stroke (DCD)</td>
<td>$5 \times 3$-state, $1 \times 6$-state, $5 \times 12$-state, $4 \times 15$-state, $4 \times 18$-state</td>
<td>2-layer (5 neurons)</td>
<td>4-layer (14, 4, 19 neurons)</td>
</tr>
<tr>
<td>TIA (DCD)</td>
<td>$3 \times 3$-state, $3 \times 6$-state, $3 \times 9$-state, $2 \times 12$-state, $2 \times 15$-state, $1 \times 18$-state</td>
<td>2-layer (5 neurons)</td>
<td>1-layer</td>
</tr>
</tbody>
</table>

Received 17 May 2023; revised 6 November 2023; accepted 24 May 2024