DeepSeer: Interactive RNN Explanation and Debugging via State Abstraction

Zhijie Wang
University of Alberta
Edmonton, AB, Canada
zhijie.wang@ualberta.ca

Yuheng Huang
University of Alberta
Edmonton, AB, Canada
yuheng18@ualberta.ca

Da Song
University of Alberta
Edmonton, AB, Canada
dsong4@ualberta.ca

Lei Ma
University of Alberta, Canada
The University of Tokyo, Japan
ma.lei@acm.org

Tianyi Zhang
Purdue University
West Lafayette, IN, USA
tianyi@purdue.edu

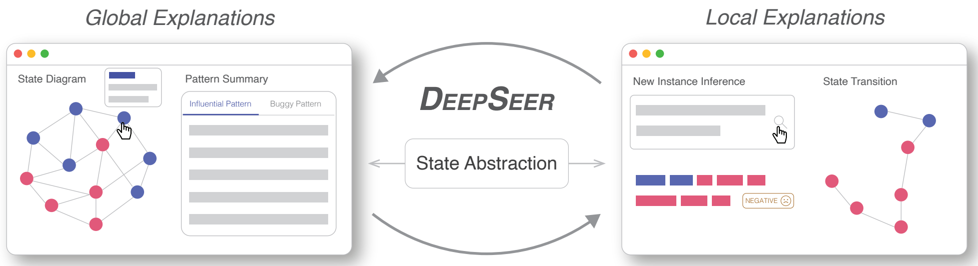


Figure 1: DeepSeer is an interactive tool for supporting RNN model understanding and debugging via state abstraction. DeepSeer helps programmers by providing *Global Explanations* and *Local Explanations* as a synergistic loop for an RNN model. Programmers can use DeepSeer to quickly understand and identify potential bugs by exploring *Global Explanations*, then zoom into *Local Explanations* to contextualize global explanations. Programmers can also debug on a specific instance according to *Local Explanations*, then validate their debugging hypothesizes by zooming out to compare with *Global Explanations*.

ABSTRACT

Recurrent Neural Networks (RNNs) have been widely used in Natural Language Processing (NLP) tasks given its superior performance on processing sequential data. However, it is challenging to interpret and debug RNNs due to the inherent complexity and the lack of transparency of RNNs. While many explainable AI (XAI) techniques have been proposed for RNNs, most of them only support local explanations rather than global explanations. In this paper, we present DeepSeer, an interactive system that provides both global and local explanations of RNN behavior in multiple tightly-coordinated views for model understanding and debugging. The core of DeepSeer is a state abstraction method that bundles semantically similar hidden states in an RNN model and abstracts the model as a finite state machine. Users can explore the global model behavior by inspecting text patterns associated with each state and the transitions between states. Users can also dive into individual predictions by inspecting the state trace and intermediate prediction results of a given input. A between-subjects user study with 28 participants shows that, compared with a popular XAI technique, LIME, participants using DeepSeer made a deeper and more comprehensive assessment of RNN model behavior, identified the root causes of incorrect predictions more accurately, and came up with more actionable plans to improve the model performance.

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**; • **Computing methodologies** → **Machine learning**; • **Software and its engineering** → **Software testing and debugging**.

KEYWORDS

Explainable AI, Model Debugging, Recurrent Neural Networks, Visualization

ACM Reference Format:
Zhijie Wang, Yuheng Huang, Da Song, Lei Ma, and Tianyi Zhang. 2023. DeepSeer: Interactive RNN Explanation and Debugging via State Abstraction. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3544548.3580852

WARNING: The toxicity detection example in the usage scenario contains some content that might be distracting.

1 INTRODUCTION

Deep neural networks (DNNs) have been increasingly adopted in practice due to their superior performance on real-world challenging tasks, e.g., self-driving [49], virtual assistant [9], and disease diagnosis [34]. The rapid development of deep learning systems brings opportunities, but also challenges and concerns. One of the major concerns arises from the interpretability of DNNs [36]. Unlike traditional software whose decision logic is manually programmed in the form of source code, a DNN model includes a large number of neurons connected by non-linear functions, whose weights are automatically learned from training data. The internal states of traditional software can be easily inspected and analyzed by setting breakpoints and checking runtime values. However, the internal states of a DNN model are high-dimensional vectors rather than symbolic values. It is hard to tell what kinds of patterns a DNN model has learned by inspecting these vectors or why the model makes a specific prediction. Therefore, this internal complexity and inscrutability of DNNs lead to significant debugging challenges, as well as concerns about the trustworthiness and reliability of DNNs.

Although there is a recently growing interest in improving the interpretability of DNNs in the ML, HCI, and Visualization communities, many existing techniques treat a DNN model as a black box and generate model-agnostic explanations such as feature importance, without revealing the inner workings of the DNN model [32, 39, 53, 54]. While there are some techniques for visualizing the hidden states in a DNN model, many of them focus on convolutional neural networks (CNNs) [8, 48, 57]. In this work, we are particularly interested in recurrent neural networks (RNNs). Compared with other kinds of DNNs, recurrent neural networks (RNNs) are capable of processing sequential data with variable lengths, such as text and audio. The recurrent architecture affords an internal memory in RNNs, which is proven effective for learning temporal patterns in sequential data. Yet this architecture also poses challenges in visualizing the internal states of RNNs. Unlike CNNs which have a fixed number of layers and neurons in each layer, RNNs are *unbounded*. Furthermore, instead of treating each layer separately, which is a common practice in CNN visualization, it is important to visualize the dynamics of RNN units, i.e., the temporal patterns embedded in a sequence of hidden states.

In this paper, we present DeepSeer, an interactive system that allows model developers to understand and debug RNN models. Our key insight is to treat an RNN model as a stateful system. By clustering and abstracting semantic similar hidden states, an RNN model can be represented as a finite-state machine (FSM), which is much smaller and more navigable compared with the original RNN model. Furthermore, instead of directly visualizing the values of hidden states as in prior work [59], DeepSeer projects hidden states to a more interpretable representation—the common words and phrases associated with these states. By inspecting the transitions among states, users can quickly identify the temporal patterns learned by the model.

To assess the overall usefulness of DeepSeer, we conducted a between-subjects user study with 28 programmers of various levels of expertise in ML and RNNs. Given a pre-trained RNN model, participants were asked to complete a model understanding task followed by a debugging task using either DeepSeer or a popular XAI tool, LIME [53]. We found that in the model understanding task, participants using DeepSeer provided more insightful answers about the model behavior, pinpointed model limitations more precisely, and gave more useful and diverse suggestions about how to improve the assigned model. Furthermore, in the model debugging task, participants using DeepSeer identified the reasons for the misclassifications more correctly than participants using LIME.

In summary, this work makes the following contributions:

- **System.** We design and develop an interactive system for understanding and debugging the internal behavior of RNNs. We first leverage the state abstraction method to abstract an RNN model as a finite state machine through bundling semantically similar hidden states. Then we design and implement three tightly-coordinated views: state diagram view, pattern summary view, and instance view to visualize and interpret the internal behavior of an RNN model from different perspectives. We have open-sourced our system on GitHub [1].
- **Visualizations and interactions.** We propose a set of visualization and interaction designs to facilitate the interpretation and debugging of RNNs at different granularities. Specifically, we combined state diagrams, responsive tooltips, state traces, color highlighting, filtering, instance matching, and pattern summarization to simultaneously show the global model behavior, instance-level explanations, critical patterns, and similar instances.
- **Evaluation.** A between-subjects user study demonstrates the usefulness of DeepSeer to ML developers when understanding the overall behavior of a model and debugging misclassifications.

2 BACKGROUND: RECURRENT NEURAL NETWORKS

Recurrent Neural Networks (RNNs) are a type of deep neural network that is specifically designed for processing sequential input, e.g., text data. In this section, we briefly introduce the basics of it.

As shown in Fig. 2, an RNN model takes sequential inputs $\{x_1, x_2, \ldots, x_T\}$. The RNN model first initializes its hidden state vector $h_0 \in \mathbb{R}^N$, where N is the dimension of this hidden state vector. At a time step t, the RNN model takes an input $x_t (1 \leq t \leq T)$ to update its internal hidden state from the last time step h_{t-1} to the new hidden state h_t. This process can also be seen as maintaining and updating the "hidden memory" of an RNN model. Therefore, to understand an RNN model, it is important to interpret such "memory" maintained in different hidden states [43, 59].

To make a prediction at time step t, an RNN model transforms the hidden state h_t into the desired output format y_t. For instance, to perform a classification task, the hidden state h_t is usually fed into an MLP (multilayer perceptron) network to project h_t into $u_t \in \mathbb{R}^K$, where K is the number of classes. Then a probability distribution p_t is computed through a "softmax" function:

$$p_t = \text{softmax}(u_t)$$
$$p_t^i = \frac{e^{u_t^i}}{\sum_{j=1}^{K} e^{u_t^j}} \quad \text{for } i = 1, \ldots, K \tag{1}$$

The prediction result at time step t is further computed by finding a label k which produces the maximum probability p_t^k.

[1]https://github.com/momentum-lab-workspace/DeepSeer

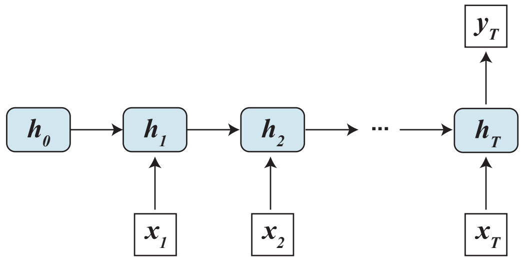


Figure 2: The workflow of a basic recurrent neural network (RNN). At each time step t, RNN takes an input x_t to update its hidden state h_t. The prediction result at time step t (y_t) is obtained by processing the hidden state h_t.

Note that the process of updating the hidden state h_t can be achieved by different types of RNN units, such as the Elman RNN cell [18], long short-term memory (LSTM) [23], and gated recurrent unit (GRU) [12]. In our user study sessions, we use GRU, which shows better efficiency compared with other variants. Note that our proposed system only requires access to the hidden states and does not require access to the updating process inside an RNN unit. Therefore, it can be generalized to different types of RNN units.

3 RELATED WORK

3.1 Explainable AI

Our work is most related to Explainable AI (XAI), since it promotes model interpretability by abstracting a DNN model as a finite state machine (i.e., a global explanation) and by rendering the state trace of a given instance (i.e., a local explanation). Previous studies have shown that supporting model interpretability can increase user acceptance and trust of the system [17, 22, 30, 55], improve fairness [14], and improve human-AI team performance [11]. A good interpretation should be in an interpretable domain [45], i.e., mapping any of abstract concepts (e.g., numeric vectors) into a domain (e.g., images, texts) that the human can understand. Our work is inspired by this principle—instead of visualizing hidden state values as in some prior work [59], we map hidden states back to linguistic patterns in the text corpus. In this way, users can easily recognize what patterns an RNN model has learned from the training data.

Existing XAI methods can be roughly grouped into two categories: model-agnostic methods and model-aware methods. Model-agnostic methods [39, 53, 54] treat the model to be explained as a black box. LIME [53] is a well-known technique in this category. Given an input instance, it learns a simpler and interpretable model (also known as a surrogate model), such as a linear regression model, to approximate a complex model using the training data near the given instance. By rendering the feature's importance in the surrogate model, LIME generates a local explanation for the prediction of the given instance. However, these model-agnostic methods usually ignore the internal behavior of a model when generating explanations. Specifically, given an RNN model, they do not take the transitions between different hidden states into account. On one side, this may lead to low-fidelity explanations [52]. On the other hand, advanced user groups such as model developers may find it insufficient to debug model behavior [59]. To address this challenge, DeepSeer is designed to investigate the internal behavior of an RNN model via a novel finite state machine abstraction.

Unlike model-agnostic methods, model-aware methods try to open up the *black box* of a DNN. Among different model-aware methods, our work is most related to attribution-based methods and influence function methods. Attribution-based methods [56, 58, 64] often use activation or gradient information in a DNN model to compute the importance score for input features, e.g., pixels in an image, tokens in a sentence. For example, Karpathy et.al. [27] presents a visualization that maps neurons' activation to individual characters in a sentence. This visualization is only applicable to individual sentences (i.e., local explanations), which becomes hard to interpret with a large number of sentences. Our work differs in a way that we aggregate words and phrases with similar hidden states from many sentences in a finite state diagram while also providing a way to delve into the state trace of individual sentences. Influence function methods [7, 31, 32] compute the influence of an individual training instance based on gradients and identify a set of instances that have a big impact on model predictions. Our design of *influential patterns* and *possible buggy patterns* draws inspirations from these methods. Specifically, DeepSeer summarizes short text patterns which usually significantly affect model predictions or have led to possible bugs by analyzing the hidden states of training data.

We further refer readers to existing surveys and literature reviews [2, 6, 44] for more details about different XAI methods.

3.2 DNN Debugging, Testing, and Repairing

Several explainable AI techniques have been used to understand and debug model errors [3, 29, 32, 53]. For example, Ribeiro et.al. conducted a user study with 27 participants and showed that the explanations generated by LIME could be used to detect spurious correlations learned by a model. Koh et.al. [32] have shown that influence functions can be used to debug domain mismatch. However, Adebayo et.al. [3] found that post-hoc model explanations, especially attribution-based methods, are sometimes ineffective for detecting certain kinds of bugs such as label error and out-of-distribution error.

In parallel, the Software Engineering (SE) community has developed several techniques by adapting traditional SE techniques to debug, test, and repair DNN models [40, 41, 62, 63]. DeepRepair [63] uses a style-transfer-based data augmentation method to repair DNN models. RNNRepair [62] identifies influential instances for retraining and remediates two types of incorrect predictions at the sample and segment levels. MODE [41] presents a debugging workflow by first conducting model state differential analysis and then selecting training instances for retraining. LAMP [40] provides data provenance information by computing the importance of input through automated differentiation. These techniques focus on automating the debugging and retraining pipeline and do not involve humans in the loop. Our work differs from these techniques in two ways. First, DeepSeer aims to provide a comprehensive understanding of model behavior by abstracting RNNs as a state diagram and identifying influential patterns, going beyond diagnosing model prediction errors. Second, to enable model developers to diagnose model errors, DeepSeer renders the intermediate prediction results of input and provides affordances for investigating how individual words and phrases influence the prediction result, rather than automatically localizing the root cause of a model error.

3.3 RNN Visualization

Many DNN visualization techniques have been proposed to help users understand and analyze the inner workings of DNN models. The most related visualization techniques to us are those specifically designed for RNNs [27, 43, 59]. Karpathy et.al. [27] visualizes which characters in an input sentence the RNN attends to based on the activation function output. Li et.al. [33] use a gradient-based salience score, rather than neuron activation, to measure the importance of each word in an input sentence. The salience scores are then visualized in a heatmap. Both visualizations are static and can only visualize the hidden states of a single input at a time. Strobelt et.al. [59] extend them by building an interactive visualization approach called LSTMVis. LSTMVis renders individual hidden states in a parallel coordinates plot. It allows users to interactively select specific segments of an input sentence and search for other inputs with similar hidden states. However, given that RNNs typically have hundreds or even thousands of hidden states, visualizing individual hidden states can lead to significant cognitive overhead for users. To address this issue, DeepSeer clusters similar hidden states to an abstract state and represents an RNN model as a finite state machine. This significantly reduces the number of states users need to keep track of and also allows DeepSeer to directly visualize the finite state machine to provide a global view of the entire model rather than individual hidden states. Our work is also related to RNNVis [43]. RNNVis clusters similar hidden states as memory chips and renders text inputs associated with each cluster as word clouds. However, unlike a finite state machine, this design does not capture the transition between hidden states or render longer linguistic patterns beyond common words. Furthermore, DeepSeer provides additional features to facilitate model inspection and debugging, e.g., rendering intermediate prediction results, summarizing influential patterns and buggy patterns, etc.

4 DESIGN GOALS AND SYSTEM OVERVIEW

In this section, we first summarize the design goals of our system based on a literature review. Then, we present a system overview to discuss how our system design supports each design goal.

4.1 User Needs and Design Goals

To understand the needs of RNN developers, we perform a literature review of previous papers that have done a formative study of interpreting DL models, have done a user study of existing tools, or have discussed the challenges and opportunities of explaining and debugging DL models. Based on the literature review, we summarize the following design goals for DeepSeer:

G1. Help users understand the overall behavior of an RNN model. Previous studies have shown that model developers prefer to have a high-level understanding of what has been learned by the model [13, 35, 43]. For instance, Kaur et al. surveyed 197 ML developers about the interpretability tool's capabilities, and 61% of responses mentioned the importance of global explanations [28]. Specifically, Ming et al. highlighted the importance of rendering the semantic information captured by the hidden states of an RNN model [43]. Thus, DeepSeer should help model developers understand the overall behavior of an RNN model, especially the semantic information learned by its hidden states.

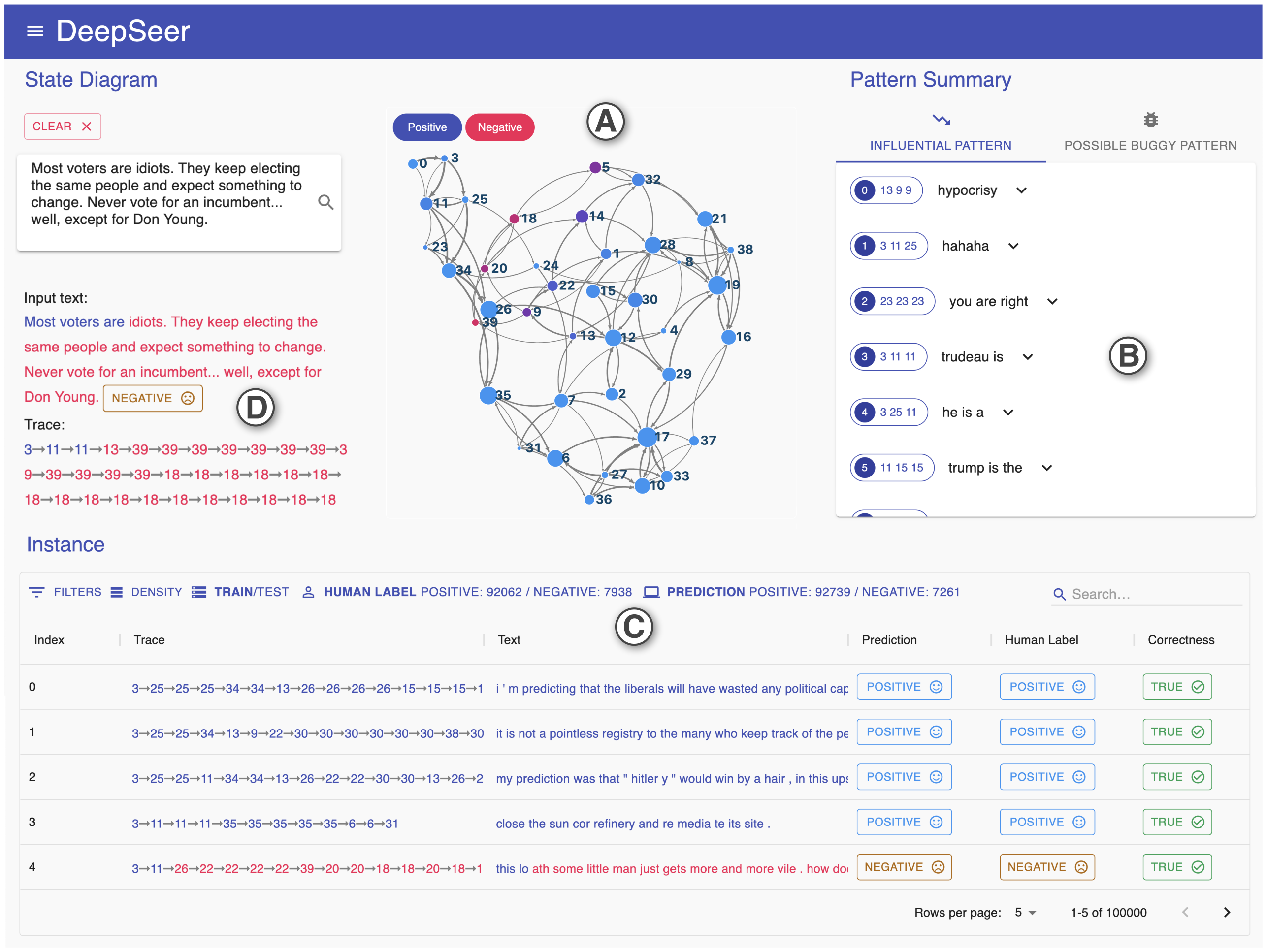
G2. Help users understand the model decision-making process on a specific input of interest. When inspecting individual prediction results, especially incorrect ones, model developers wish to understand why the model makes such a prediction on the particular input [5, 24, 35]. For instance, through a formative study with nine ML developers, Hohman et al. [24] found that users wanted to see how different features contributed to the model's decision. Furthermore, Kahng et al. interviewed fifteen Facebook developers and found that a natural way for them to understand complex models was to inspect the model behavior on individual examples [26]. The importance of local explanation is also confirmed by the large-scale survey [28]—65% of respondents considered local explanations important.

G3. Help and assist users in searching for similar data. Through a user-centered design process with two NLP developers, Liu et al. found that NLP developers typically follow an "exploration-centric" approach to discover and debug errors in an NLP model [38]. That is, developers prefer to inspect and compare predictions among similar input examples to get insights. Therefore, traceability should also be provided to help users easily explore similar examples when debugging a model prediction [24]. Specifically, Strobelt et al. highlighted that matching similar examples for RNN could help developers validate an interpretation hypothesis [59].

G4. Help users summarize the common characteristics of input data. Inspecting individual data points can be tedious and time-consuming, hindering insight discovery. Kahng et al. found that model developers at Facebook often curated subsets of data with common characteristics to understand how a model behaves at high-level categorization [26]. Furthermore, helping users identify common input characteristics is particularly useful for error analysis. Jin et al. found that ML developers usually needed to examine the characteristics shared by a set of wrong predictions and verify whether error patterns formed by these characteristics make sense [25]. However, this is often manually done by users based on their domain knowledge. Therefore, DeepSeer should support users in identifying and examining common characteristics of inputs, especially mispredicted inputs.

4.2 System Overview

To support users gaining a high-level understanding of what has been learned by an RNN model (**G1**), we choose to render an RNN model as a state diagram in which each node is a group of similar hidden states from the RNN model, as shown Figure 3 Ⓐ. Compared with the original RNN model, which has hundreds or thousands of hidden states, the state diagram is much smaller after state clustering and thus more navigable. Furthermore, DeepSeer binds each node with the text patterns memorized by the corresponding hidden states to help users interpret the semantic meaning of the hidden states. Compared with an alternative design of directly visualizing the hidden states values [27, 59], which are high-dimensional arrays and hard to interpret, the state diagram is easier to navigate and inspect.



CLEAR ✕

Most voters are idiots. They keep electing the same people and expect something to change. Never vote for an incumbent... well, except for Don Young.

Input text:
Most voters are idiots. They keep electing the same people and expect something to change. Never vote for an incumbent... well, except for Don Young. NEGATIVE ☹

Trace:
3→11→11→13→39→39→39→39→39→39→39→39→39→39→39→39→18→18→18→18→18→18→18→18→18→18→18→18→18→18→18→18→18

0	13 9 9	hypocrisy
1	3 11 25	hahaha
2	23 23 23	you are right
3	3 11 11	trudeau is
4	3 25 11	he is a
5	11 15 15	trump is the

FILTERS DENSITY TRAIN/TEST HUMAN LABEL POSITIVE: 92062 / NEGATIVE: 7938 PREDICTION POSITIVE: 92739 / NEGATIVE: 7261 Search…

Index	Trace	Text	Prediction	Human Label	Correctness
0	3→25→25→25→34→34→13→26→26→26→26→15→15→15→1	i ' m predicting that the liberals will have wasted any political cap	POSITIVE ☺	POSITIVE ☺	TRUE ✓
1	3→25→25→34→13→9→22→30→30→30→30→30→30→38→30	it is not a pointless registry to the many who keep track of the pe	POSITIVE ☺	POSITIVE ☺	TRUE ✓
2	3→25→25→11→34→34→13→26→22→22→30→30→13→26→2	my prediction was that " hitler y " would win by a hair , in this ups	POSITIVE ☺	POSITIVE ☺	TRUE ✓
3	3→11→11→11→35→35→35→35→35→6→6→31	close the sun cor refinery and re media te its site .	POSITIVE ☺	POSITIVE ☺	TRUE ✓
4	3→11→26→22→22→22→22→39→20→20→18→18→20→18→1	this lo ath some little man just gets more and more vile . how do	NEGATIVE ☹	NEGATIVE ☹	TRUE ✓

Rows per page: 5 1-5 of 100000

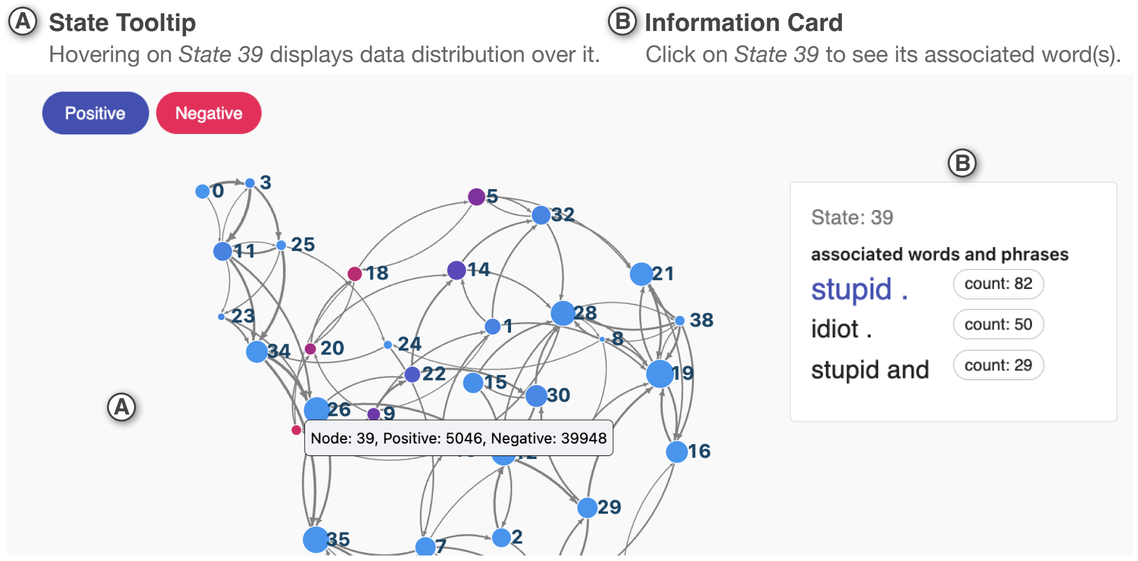
Figure 3: DeepSeer, an interactive system for visualizing, understanding, and debugging RNN models. (A) The *State Diagram View* displays the abstracted states and transitions of an RNN model. (B) The *Pattern Summary View* displays common text patterns learned by an RNN model. (C) The *Instance View* displays the raw data as an interactive data grid for users to explore data used to train or test an RNN model. (D) *Intermediate Prediction Results* are rendered when users input a new sentence.

To help users understand the model decision-making process on specific inputs (**G2**), DeepSeer visualizes the intermediate model prediction result after an RNN model reads each word in an input sentence (Figure 3 Ⓓ). In this way, users can easily see which word sways the decision of the model and contributes more to the final result. To support **G3**, DeepSeer allows users to search input sentences with *similar text patterns* (i.e., have the same keyword or follow the same regular expression) or with *similar model behavior pattern* (i.e., have the same state or follow the same state trace) in an instance view (Figure 3 Ⓒ). To help users find common patterns (**G4**), DeepSeer proactively identifies frequent text patterns that have a high influence on model prediction results, as well as patterns that are shared among incorrect predictions (Fig. 3 Ⓑ). Such common patterns can also serve as a complementary global explanation method (**G1**), since it provides more straightforward starting points for investigation if users find a state diagram overwhelming.

5 DESIGN AND IMPLEMENTATION

5.1 State Abstraction

To generate a state diagram from an RNN model, we develop a method that clusters semantically related hidden states of the RNN model into an abstract state. Our work is inspired by the model-based analysis of stateful RNNs [15, 16, 47, 51, 61]. These works apply various techniques to extract interpretable state transition models (e.g., discrete-time Markov chain, automata) from stateful RNNs. By turning complex RNNs into interpretable state transition models, black boxes are turned into more transparent models and thus improve the model interpretability, which also provides the possibility for further analysis. We choose to build on top of a state-of-the-art method, DeepStellar [16], since it is demonstrated to be effective in various tasks, including adversarial detection [16], DNN testing [16], and DNN repair [62]. Previous work has also shown that abstracted states can make the same prediction as the original RNN model in 97% of test data [62].


Ⓐ State Tooltip
Hovering on *State 39* displays data distribution over it.
Ⓑ Information Card
Click on *State 39* to see its associated word(s).
Positive
Negative
Ⓐ
Ⓑ
0
3
25
11
23
34
20
24
18
14
5
32
21
1
28
8
38
19
22
15
30
26
9
16
29
35
7
2

Node: 39, Positive: 5046, Negative: 39948

State: 39

associated words and phrases

stupid .	count: 82
idiot .	count: 50
stupid and	count: 29

Figure 4: Interacting with the *State Diagram View* to understand the abstracted states of an RNN model.

To obtain the FSM representation for a trained RNN, we abstract over both the states and the transitions. Appendix A presents the algorithm for state abstraction. Here we briefly summarize how it works. For each instance in the training data, our method first records the intermediate hidden vectors $\{h_1, h_2, \ldots, h_l\}$ during inference, where h_i ($1 \leq i \leq l$) is a concrete hidden state of an RNN model. l denotes the number of tokens in a sentence. Our method then applies Principle Component Analysis (PCA) for dimension reduction on all concrete states collected from training data before abstraction. Different from DeepStellar [16], which uses an interval-based method for states abstraction, our method applies Gaussian Mixture Model (GMM) [42] to cluster similar concrete states. GMM addresses two key limitations in the interval-based method: 1) newly generated hidden vectors can fall outside the interval at test time, resulting in unknown states; 2) the number of states grows exponentially with k dimension and m intervals (m^k), and too many states can be hard to visualize. With state abstraction, the model prediction process on a given input can be modeled as a sequence of abstract states. We call this state sequence the *trace* of model prediction.

We conducted a quantitative analysis of the faithfulness of state abstraction. We measured the prediction consistency between the abstracted and original models in the three different NLP tasks from the usage scenario (Section 6) and the user study (Section 7.2). The prediction consistency on the test data is 99%, 97%, and 85%, respectively. This implies that abstracted models can faithfully represent the behavior of RNNs. Appendix B includes the experiment details.

Different from the previous work focusing on state abstraction technique itself or using the technique for model testing and repairing, our work is the first to extend this technique for interactive model explanation and debugging with a more accessible user interface. Our work integrates state abstraction into a "human-in-the-loop" approach for the first time to support users in understanding and debugging an RNN model with rich interaction mechanisms. In the following subsections, we will introduce the interactive features of DeepSeer built on top of state abstraction.

5.2 State Diagram View

The *State Diagram View* visualizes the finite state machine that is abstracted from the given RNN in the previous step. It provides an overview of the model behavior and helps users understand the semantic meanings of its hidden states. Users can navigate through different state nodes to explore what prediction result a state often leads to and how many times this state has been visited. Specifically, each state is color-coded based on how frequently the input instances going through this state have a specific prediction result. The size of a state node is determined by how many input sentences have visited this node during the training time. For example, in Fig. 4 Ⓐ, since the RNN model makes two possible predictions—positive comment or negative comment, all nodes are assigned to two distinct colors—blue for positive comments and red for negative comments. Since there are fewer red nodes and the red nodes have a much smaller size than blue nodes, one can interpret that the training dataset has more positive comments than negative comments and the RNN model is more likely to make a positive prediction. The width of an edge between two states indicates how frequently this transition has occurred during the training time. The RNN model moves from one state to another state when it reads more words from a given input sentence.

When a user hovers the mouse over a state, a tooltip is rendered to provide more details about this state, e.g., the number of training instances that go over this state and is eventually predicted to a specific result (Fig. 4 Ⓐ). When users click on a state, an information card (Fig. 4 Ⓑ) popped up showing the phrases and words that are frequently associated with this state in the training data. This feature allows users to interpret the semantic information memorized by hidden states. Clicking on a state also updates the

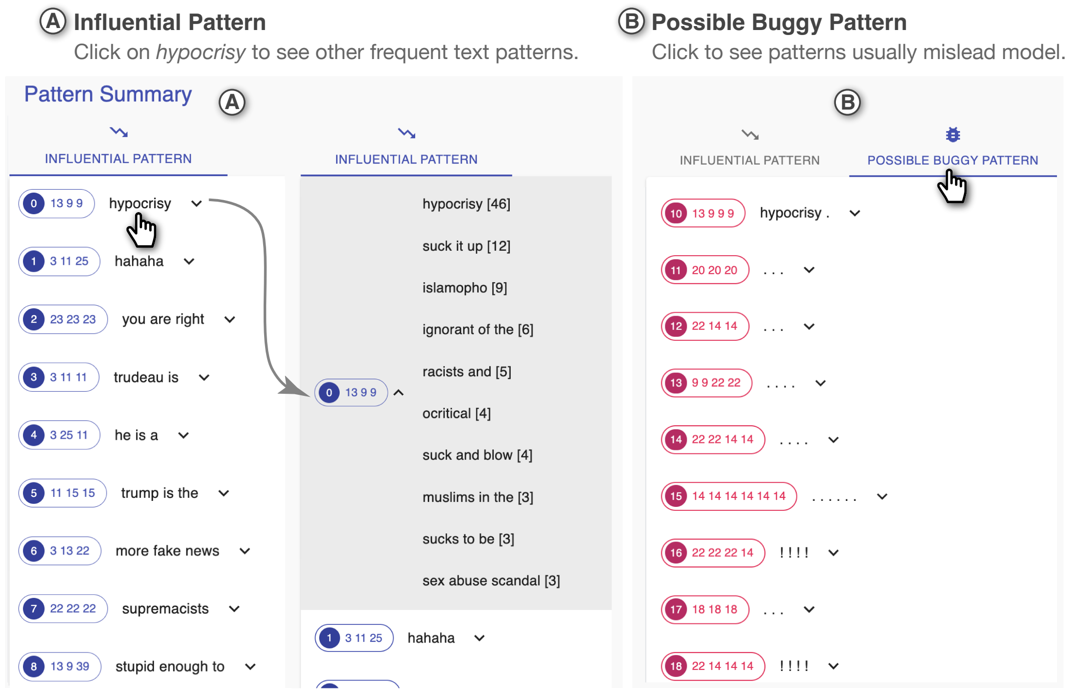


Figure 5: Interacting with the *Pattern Summary View* in DeepSeer to inspect the text patterns learned by an RNN model.

instance view to filter out the input instances that do not visit this state during model prediction.

5.3 Pattern Summary View

The *Pattern Summary View* renders common patterns based on frequent state transitions during the training time. Basically, a frequent subsequence of states is viewed as a pattern, which can be further converted into a sequence of words based on the words or phrases associated with each state. DeepSeer identifies two kinds of patterns: *Influential Patterns* and *Possible Buggy Patterns*.

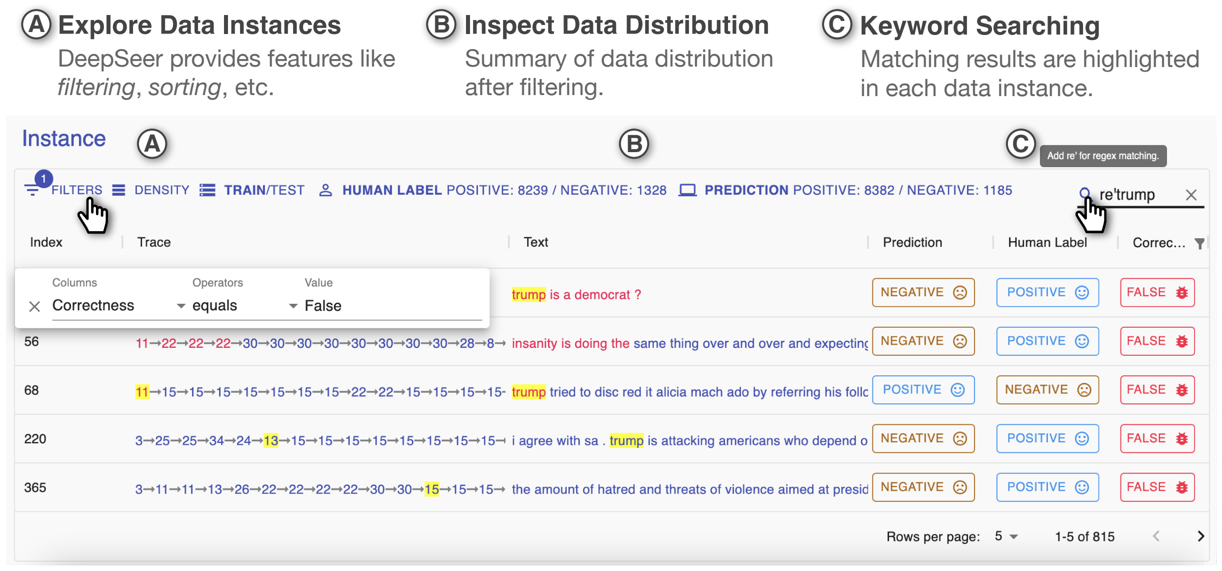
Influential Patterns are patterns that change the model's intermediate predictions, e.g., changing from a positive comment to a negative comment after reading certain words in the middle of a given input sentence. To identify influential patterns learned from the training data, DeepSeer first identifies the pivoting points (i.e., the point where the intermediate prediction changes) in the state trace of each training instance. Then DeepSeer splits each state trace into subsequences based on the pivoting points. These subsequences are sorted based on their frequency and rendered in a descending order in the pattern summary view.

Possible Buggy Patterns are mined only from incorrectly predicted instances from the training data. These patterns indicate the cases where the RNN model does not learn well and thus makes a misprediction. To identify buggy patterns, DeepSeer first divides the training data S into two subsets according to the correctness of their prediction results. Let's denote the subset only include correct predictions as S_c and the subset only include false predictions as S_f, respectively. Then we use TKS [19] (Top-K Sequential pattern mining) to mine frequent subsequence patterns from each subset. A subsequence pattern is considered possibly buggy if it only occurs in the misclassified inputs (S_f), not in the correctly classified inputs (S_c). These buggy patterns are sorted based on their frequency and rendered in a descending order in the pattern summary view.

Users can click on a specific pattern to see the top frequent phrases associated with this pattern (Fig. 5 Ⓐ). This pattern summary view allows users to know what patterns the model has learned, and how these patterns would affect the model's predictions. Furthermore, *Possible Buggy Patterns* allows users to recognize potential prediction risks of an RNN model. Clicking on a pattern will update the instance view to filter out data instances that do not follow the selected pattern.

5.4 Instance View

The *Instance View* (Fig. 6) is a scrollable data grid of the raw data used to train or test the model. The rows of the data grid are individual data instances, and the columns are: *Index*, *State Trace*, *Text*, *Prediction*, *Human Label*, and *Correctness* of the data instance. Users can sort and filter the data instances by each column (Fig. 6 Ⓐ). Users can use the TRAIN/TEST tab to switch between training and test instances. The distributions of human labels and model prediction results are summarized and rendered on top of this view (Fig. 6 Ⓑ). As users filter the data instances, these distributions are also updated accordingly. Users can also search for specific input data based on keywords or regular expressions (Fig. 6 Ⓒ). The matched results will be *highlighted* for better visualization. For each instance, its words and states are colored based on the intermediate prediction results. Clicking on a row in the instance view


Ⓐ Explore Data Instances
DeepSeer provides features like *filtering*, *sorting*, etc.

Ⓑ Inspect Data Distribution
Summary of data distribution after filtering.

Ⓒ Keyword Searching
Matching results are highlighted in each data instance.

Instance

Add re' for regex matching.

FILTERS DENSITY TRAIN/TEST HUMAN LABEL POSITIVE: 8239 / NEGATIVE: 1328 PREDICTION POSITIVE: 8382 / NEGATIVE: 1185

re'trump

Columns	Operators	Value
Correctness	equals	False

Index	Trace	Text	Prediction	Human Label	Correc...
		trump is a democrat ?	NEGATIVE	POSITIVE	FALSE
56	11→22→22→22→30→30→30→30→30→30→30→30→28→8→	insanity is doing the same thing over and over and expecting	NEGATIVE	POSITIVE	FALSE
68	11→15→15→15→15→15→15→15→22→22→15→15→15→15-	trump tried to disc red it alicia mach ado by referring his follc	POSITIVE	NEGATIVE	FALSE
220	3→25→25→34→24→13→15→15→15→15→15→15→15→15→	i agree with sa . trump is attacking americans who depend o	NEGATIVE	POSITIVE	FALSE
365	3→11→11→13→26→22→22→22→22→30→30→15→15→15→	the amount of hatred and threats of violence aimed at presid	NEGATIVE	POSITIVE	FALSE

Rows per page: 5 1-5 of 815

Figure 6: Interacting with the *Instance View* in DeepSeer to explore the data used to train/test an RNN model.

will update the state diagram view to render the state transitions of the selected data instance.

5.5 Intermediate Prediction Results

The previous sections describe how users can use different views to achieve an overall understanding of the model behavior. Previous studies have shown that it is also important to allow users to inspect and debug model predictions on individual instances [39, 53, 56]. To support instance-level inspection and debugging, DeepSeer allows users to enter an input sentence in the text box in Fig. 7. After clicking on the magnifier button, DeepSeer renders the state trace of the model prediction on the given input. Furthermore, each word in the input sentence is colored based on the *intermediate prediction result*. For example, in Fig. 7 Ⓑ, "red" and "blue" indicate negative comments and positive comments respectively. RNN typically uses the final hidden state after reading the entire sentence to compute the class probabilities. In DeepSeer, the hidden state after reading each word in a sentence is fed into the output layer to generate intermediate predictions. Through these intermediate predictions, users can inspect how the prediction result has changed over time as the RNN model reads more words in the input sentence. The pattern summary view is also updated with only influential patterns and possible buggy patterns related to the given input sentence. With these supports, users can quickly find suspicious words or phrases when debugging an incorrect model prediction.

6 USAGE SCENARIO

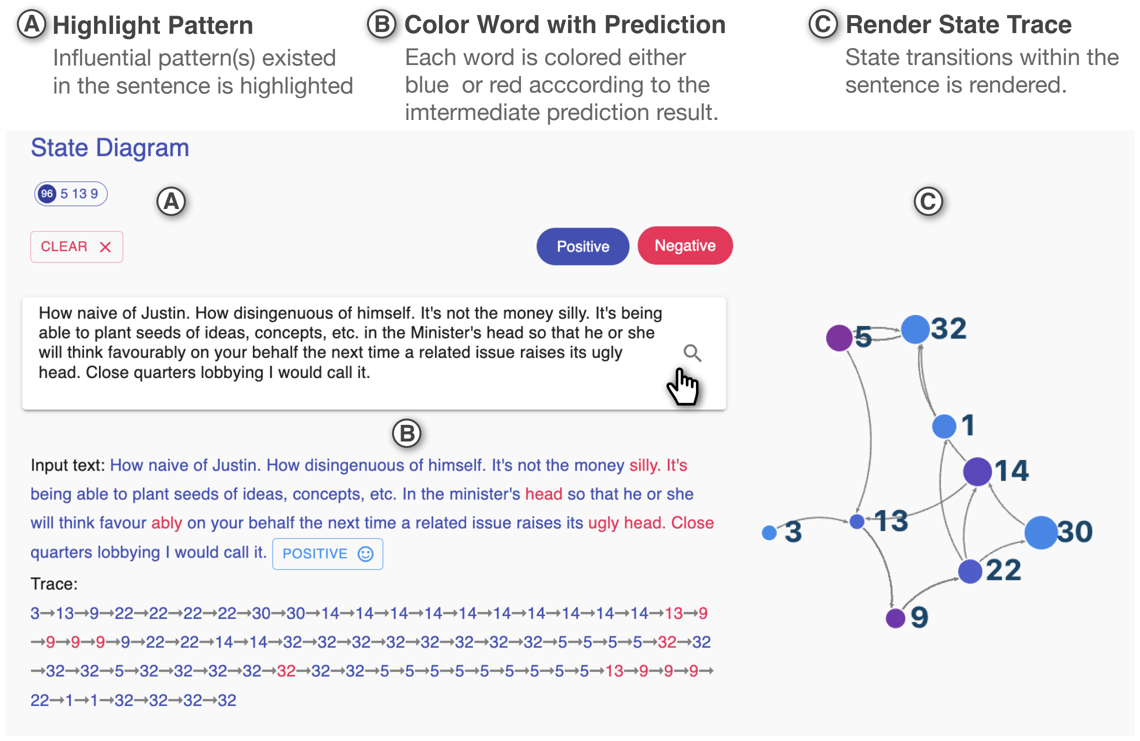
Suppose Alice is a model developer, and she trains a toxicity detection RNN model using the Toxic dataset [2]. This model predicts whether a sentence has a positive tone or negative tone. Her model achieves 95% accuracy on the training data but only 89% on the test data. Alice uses DeepSeer to figure out why there is such a performance drift.

6.1 Visualizing and Understanding an RNN Model

Alice first attends to the *state diagram view*, which gives her an overview of the trained RNN model as a finite state machine (Figure 3 Ⓐ). In this state diagram, each node represents a group of similar hidden states in the RNN model. A blue node indicates that the model is more likely to give a Positive intermediate prediction after visiting this state, while a red node indicates that the model is more likely to give a Negative intermediate prediction. Alice hovers her mouse over a red node named State 39. As shown in Figure 4, a tooltip then pops up showing that the model makes a negative intermediate prediction 39,948 times while only 5,046 times for positive ones, after visiting this state. When Alice clicks on the node of State 39, an information card is displayed on the right (Figure 4 Ⓐ), showing common words and phrases associated with the state, such as "stupid", "idiot", and "stupid and". Alice glances over several sentences with these words in the *instance view* below (Figure 3 Ⓒ) to check whether they are *hate comments*. In this way, Alice confirms that her RNN model indeed learns some meaningful patterns from the training data.

While inspecting text patterns associated with each state is helpful, Alice finds it cumbersome to check all states in the state diagram. So she switches to the *pattern summary view* (Fig. 3 Ⓑ) to understand the model from another perspective. This view shows text patterns that have a significant impact on the model prediction (i.e., influential patterns). Alice finds some interesting patterns such as "more fake news" and "stupid enough to". When Alice clicks on one of the patterns, "hypocrisy" (Fig. 5 Ⓐ), it is expanded to show a list of other frequent patterns that are associated with the same

[2]https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification


Ⓐ Highlight Pattern
Influential pattern(s) existed in the sentence is highlighted
Ⓑ Color Word with Prediction
Each word is colored either blue or red acccording to the imtermediate prediction result.
Ⓒ Render State Trace
State transitions within the sentence is rendered.
State Diagram
96 5 13 9
Ⓐ
CLEAR ✕
Positive
Negative
Ⓒ
Ⓑ
5
32
1
14
3
13
30
22
9

How naive of Justin. How disingenuous of himself. It's not the money silly. It's being able to plant seeds of ideas, concepts, etc. in the Minister's head so that he or she will think favourably on your behalf the next time a related issue raises its ugly head. Close quarters lobbying I would call it.

Input text: How naive of Justin. How disingenuous of himself. It's not the money silly. It's being able to plant seeds of ideas, concepts, etc. In the minister's head so that he or she will think favour ably on your behalf the next time a related issue raises its ugly head. Close quarters lobbying I would call it. POSITIVE

Trace:
3→13→9→22→22→22→22→30→30→14→14→14→14→14→14→14→14→14→14→13→9→9→9→9→9→22→22→14→14→32→32→32→32→32→32→32→32→5→5→5→5→32→32→32→32→5→32→32→32→32→32→32→32→5→5→5→5→5→5→5→5→5→13→9→9→9→22→1→1→32→32→32→32

Figure 7: ***Intermediate prediction results*** **help users interpret model's prediction when debugging.**

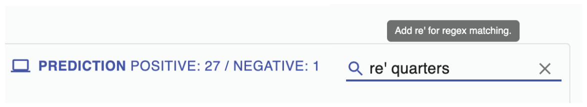
state sequence, ⑬ → ⑨ → ⑨, sorted by frequency.[3] For example, this state sequence also memorizes "suck it up" (12 sentences), "ignorant of" (6 sentences), and "suck and blow" (4 sentences) in the training data. As Alice clicks on each pattern, the *instance view* is also updated to filter training and test data that does not follow the clicked pattern. Alice is also curious about which text patterns may cause incorrect model predictions. So she switches to the list of possible buggy patterns. These buggy patterns are summarized from misclassified sentences only, rather than the entire training dataset. Alice sees some patterns such as "...", "!!!", and "???" (Fig. 5 Ⓑ). It seems that her model learns some spurious correlations between punctuation and prediction results, which may have contributed to many errors. To prevent the model from learning these spurious correlations, Alice plans to remove this punctuation to clean the training data, which may lead to better model performance.

6.2 Debugging an RNN Model

Now Alice wants to dig into the data and investigates why some sentences are misclassified after having a high-level understanding of the model behavior. So she turns to the *instance view* (Figure 3 Ⓒ), which shows all training and test data in a paginated table.

Alice first notices that the training data is not balanced. There are significantly more positive sentences (92062) than negative ones (7938). Alice then switches to the test data and finds misclassified sentences using the filtering feature on the *Correctness* column (Figure 6 Ⓐ). She copied a misclassified sentence to the text box and run the instance-level model explanation feature on it (Figure 7). Each word in the sentence is colored based on the intermediate prediction result. A state trace is also rendered below.

Alice quickly notices a few words that her RNN model considers negative during the prediction, such as "ugly head." Even though such insulting words have been recognized by the model, this sentence is eventually predicted positive. It seems the model quickly forgets these insulting words after seeing the subsequent words in the sentence. For example, after seeing "quarters", the intermediate prediction changes from negative to positive.



PREDICTION POSITIVE: 27 / NEGATIVE: 1

re' quarters

Figure 8: Alice search for "quarters" in the training data.

To verify this hypothesis, Alice searches sentences that contain "quarters" in the training set using the keyword search feature in the *instance view*. Alice finds 27 positive sentences and only 1 negative one that contains "quarters" in the training set (Fig. 8). Since many sentences with "quarters" are positive, the model may have learnt a spurious correlation between "quarters" and the positive sentiment. Alice further confirms her hypothesis by looking at the corresponding state, State 22, associated with the word "quarters".

[3]This single word, "hypocrisy", is associated with a sequence of three states, since it is tokenized into three tokens (hypo-, -cri-, -sy) in the training set, each of which is bound to one state. Such a tokenization mechanism is widely used in NLP to address out-of-vocabulary issues.

Alice finds that State 22 is a positive state. Therefore, Alice concludes that one reason for this misclassification is data imbalance, where most sentences with "quarters" are labelled as positive. To address this, Alice believes that one possible solution is to collect more data with this keyword to balance her training set and then re-train her model. Furthermore, given that the model quickly forgets an insulting word, Alice also plans to experiment with the long short-memory (LSTM) architecture with the attention mechanism, which can handle information in the memory for a longer time compared with a vanilla RNN.

7 USER STUDY

We conducted a between-subjects study with 28 participants to evaluate the effectiveness and usability of DeepSeer. We used LIME [53], a well-known tool for interpreting and debugging machine learning models, as a comparison baseline. Though LIME is not specialized for RNNs, it is a widely-used tool to understand and debug models. It has 10.3K stars and 1.7K forks on GitHub [4], and its Python package has been downloaded 16M times on PyPI [5]. Therefore we choose it as a more realistic baseline. Given a model prediction, LIME can generate an explanation with importance scores for elements in the input data (e.g., words in an input sentence). To enable a fair comparison, we built an interface for LIME similar to DeepSeer. The interface includes the existing visualizations provided by LIME and also includes the Instance View as in DeepSeer. It does not include the state diagram view and the pattern summary view, which are the novel contributions of DeepSeer. We investigated the following research questions to assess the overall usefulness of DeepSeer compared with LIME:

- RQ1: To what extent does DeepSeer enhance users' understanding of an RNN model compared with a commonly used model explanation and debugging tool?
- RQ2: To what extent does DeepSeer improve the accuracy of identifying the root cause of a misprediction of an RNN model compared with a commonly used model explanation and debugging tool?

7.1 Participants

We recruited 28 participants (5 female and 23 male) through several graduate student mailing lists of the CS department and the ECE department at the University of Alberta.[6] All participants had at least basic machine learning background. 15 participants were Ph.D. students, and the rest were Master's students. 23 participants had 2-5 years of machine learning experience, 3 participants had more than 5 years, and 2 participants had about 1 year. Regarding their RNN experience, 9 participants had more than 2 years of experience, 7 participants had 1 year, and 12 participants had less than 1 year. Participants also self-reported their familiarity with developing RNN models in a 7-point Likert scale question. The median is 5, with 1 referring to "*I have only heard about RNNs but never used it*" and 7 referring to "*I'm able to build an RNN model by myself.*" 25 participants said they had not used any debugging tools for DL, while 3 participants said they have used Tensorboard [1]. The studies were conducted on Zoom. Both DeepSeer and LIME were deployed as web applications that participants could access from their personal computers.

7.2 RNN Models

Since DeepSeer is designed for visualizing and debugging RNN models, we trained two RNN models for two popular ML tasks. For each RNN model, the dimension of a hidden state vector is 256. The first ML task is to predict whether a question asked on Quora is sincere or insincere. It is originally from a featured competition from Kaggle, a popular online machine learning and data science community [50]. In this task, our RNN model is trained on 100,000 Quora questions, each of which is labeled as sincere or insincere. The training accuracy of this RNN model is 93.93%, and the test accuracy is 89.07%. The second ML task is to predict the topic of a news article from a news corpus called AG's News [20]. This task is a well-known benchmark for topic classification research [65]. In this task, our RNN model is trained on 109,886 news articles labeled into four news topics, including "Sports", "Business", "World", and "Science and Technology." The training accuracy of this RNN model is 91.57%, and the test accuracy is 87.68%. DeepSeer abstracts each RNN model into 40 states. This number is decided empirically to achieve a good balance between accuracy and the cognitive effort of inspecting a state diagram. We further provide a faithfulness analysis of the abstracted model in Appendix C. During a user study session, we randomly assigned one of the two RNN models to a participant to finish the model understanding and debugging tasks. Interface of DeepSeer for each task can be found in Appendix D.

7.3 Protocol

We design a between-subjects user study where users experience one condition and one RNN model in each study session. We choose a between-subjects design rather than a within-subjects design since experiencing one condition takes around 60 minutes. Experiencing two conditions in a within-subjects design would require 120 minutes, which is too long and can lead to significant fatigue and frustration. At the beginning of each session, we asked the participants for their permission for recording. Given that this was a between-subjects study, each participant was assigned to only one RNN model in one condition. In each session, participants were only allowed to use the given tool: DeepSeer in the experiment condition or LIME [53] in the control condition. The assigned RNN models and conditions were counterbalanced across participants. At the beginning of each study session, participants were asked to first watch a 5-min tutorial video of the assigned tool, and then spend 5 minutes familiarizing themselves with the tool. Then, participants were given 30 minutes to use the assigned tool to explore an assigned RNN model and share their understanding of the model behavior through a questionnaire. The questionnaire included three main questions: (1) *What insights have you got about the model's performance and behavior?* (2) *Did you find any bugs or limitations of the RNN model? If yes, what kind of bugs have you found?* (3) *How will you further improve the model?* After filling out the model understanding questionnaire, participants were given 10 minutes to debug 5 incorrect model predictions using the assigned tool. For each incorrect prediction, they were asked to write down why the

[4]https://github.com/marcotcr/lime
[5]https://pepy.tech/project/lime
[6]This human-participated study is approved by the university's research ethics office.

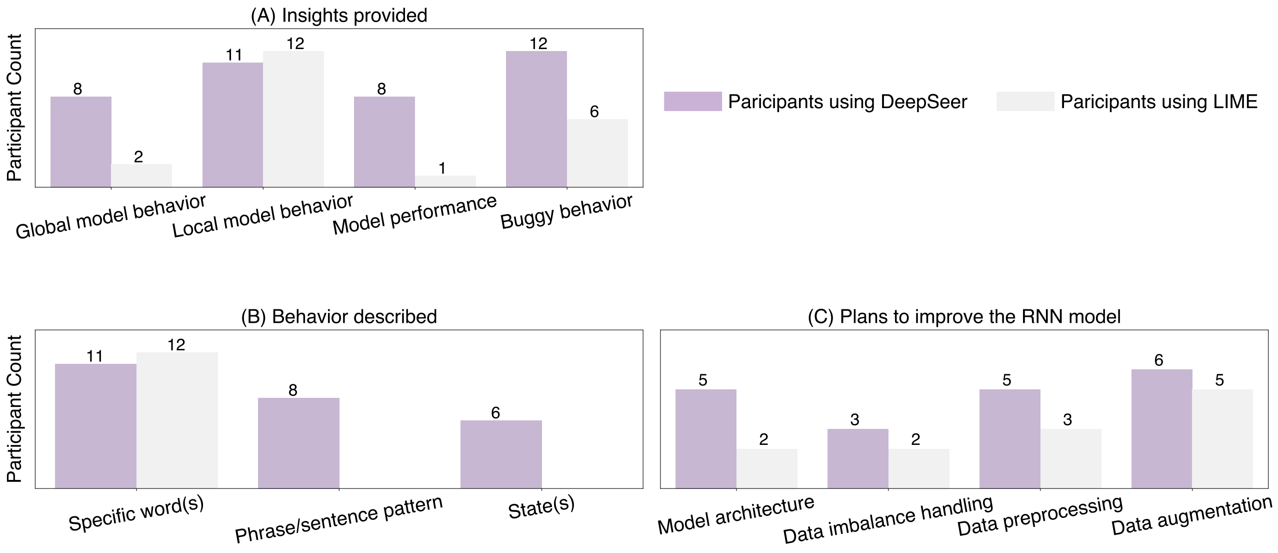


Figure 9: Participants' performance on model understanding. (A) The number of participants who mention different aspects of model behavior. (B) The number of participants who describe model behavior in different ways. (C) The number of participants who make different suggestions on model improvement.

input data was misclassified and submit their answers through a questionnaire. At the end of the study session, participants were asked to fill out a survey to share their experiences. In particular, the post-study survey included the NASA Task Load Index (TLX) questions [21] to measure the cognitive load of the study. Each participant received a $25 Amazon gift card as compensation for their time.

8 USER STUDY RESULTS

This section describes the results of the between-subjects user study. We first present and analyze participants' performance differences on model understanding and debugging tasks when using DeepSeer and LIME. Then we present participants' perception on DeepSeer's tool features as well as cognitive load. For brevity, we use P1-P14 to denote the participants using DeepSeer, and P15-P28 to denote the participants using LIME [53].

8.1 RQ1: User Performance on Model Understanding

To evaluate user performance on model understanding, two authors manually assessed and coded participants' responses and counted the number of correct insights about model behavior shared by participants. Specifically, these two authors had 4 meetings to develop a codebook and resolve labeling inconsistencies. Eventually, 651 codes were generated and categorized into 32 themes. The final Cohen's Kappa score is 0.9061. Note that one insight is considered correct only if both two authors agree.

Overall, participants using DeepSeer provided more insights (53 vs. 21) than participants using LIME. The mean difference of insights provided per participant (2.3) is statistically significant (Welch's t-test: $p = 0.0003$). Fig. 9 provides a breakdown of different kinds of insights shared by participants. Participants using DeepSeer shared much more insights about *global model behavior*, *model performance*, and *buggy behavior*. For instance, P12 said, "*It looks like the model is placing a lot of weight in the latter half of an input sentence.*" P9 wrote, "*this model is often confused by the Business and Science categories.*" Furthermore, participants using DeepSeer often referred to text patterns and states when describing model behavior, while participants using LIME mostly referred to specific keywords. P25 said, "*it is not easy to summarize patterns [with LIME] when the size of dataset is large and there are many classes.*" Since LIME is designed for local explanations, it is not surprising that only 2 participants using LIME were able to derive global explanations for an assigned RNN model.

Participants using DeepSeer also provided more useful and diverse suggestions about model improvement compared with LIME users (Fig. 9 (C)). Note that a suggestion is considered useful if it is related to the root causes of observed model misprediction and is accepted as an effective model improvement mechanism in the ML community. In particular, 5 participants using DeepSeer noticed the error pattern of forgetting previous tokens after reading more tokens and suggested adding an attention layer, while only 2 participants using LIME noticed this. P10 wrote, "*For many false predictions, the model is likely to give the right prediction at the beginning, but then turns to the wrong direction. Probably we could reduce the length of temporal dependencies with something like the attention mechanism.*" Finally, participants using DeepSeer spent 27 min 53 s (σ = 2 min 44 s) on average, while participants using LIME spent 28 min 19 s (σ = 3 min 15 s). We do not observe a significant difference in task completion time.

Table 1: The average of responses shared by participants in the model debugging task.

	Quora		AGNews		Overall	
	DeepSeer	LIME	DeepSeer	LIME	DeepSeer	LIME
Reasonable explanations provided per participant	4.3	2.7	4.3	1.1	4.3	1.9
Fault-inducing keywords mentioned per participant	5.3	2.7	4.6	1.4	4.9	2.1
Non-fault-inducing keywords mentioned per participant	0.8	3.7	1.0	3.3	0.9	3.5

8.2 RQ2: User Performance on Model Debugging

To evaluate user performance on the model debugging task, we counted the number of reasonable explanations provided by participants over five misclassified sentences. To assess the correctness of participants' answers, two authors first manually inspected the hidden states of the RNN and also the training data to diagnose the five misclassifications. Their investigation results were used as the ground-truth misclassification explanations. Then, they checked whether the participants' explanations were consistent with the ground truth.

As shown in Table 1, participants using DeepSeer provided more reasonable explanations for misclassification. Participants using DeepSeer provided 4.3 reasonable explanations for 5 misclassified sentences on average, while participants using LIME only provided 1.9 reasonable explanations. The mean difference of 2.4 is statistically significant (Welch's t-test, $p < 0.0001$).

Furthermore, we counted the number of correct fault-inducing keywords mentioned by participants. Participants using DeepSeer identified more correct fault-inducing keywords than those using LIME (mean: 4.9 vs. 2.1). The mean difference of 2.8 is statistically significant (Welch's t-test, $p < 0.0001$). In addition, participants using LIME misrecognized more keywords (mean: 3.5 vs. 0.9) as fault-inducing keywords (Welch's t-test, $p < 0.0001$). This is because LIME first learns a surrogate sparse linear model to simulate the RNN model and then computes word importance based on the linear model. This sometimes leads to unreliable explanations. Some participants also noticed this during the study. P22 commented, "*in some cases, I found that the tool [LIME] did not generate a reliable explanation.*"

One interesting observation is that participants using DeepSeer were capable of identifying more complex error patterns beyond word patterns. For example, P9 answered, "*At the very beginning, 'football' indicates the model to predict sports, which is exactly what the model does. But when 'UK' appears, the state transits to 'world' [related state] and got stuck there.*" None of the LIME users provided such insights, since LIME treats individual words separately and cannot capture the dynamics of the model's decision process.

Finally, participants using DeepSeer completed this task in an average of 9 min 9 s (σ = 1 min 21 s), while participants using LIME took an average of 9 min 25 s (σ = 0 min 11 s). There is no significant difference in task completion time.

8.3 User Perception and Cognitive Load

Our post-study survey solicited participants' feedback on all key features of DeepSeer. Overall, participants considered DeepSeer's visual encoding and interface intuitive, helpful, and clear. Among 14 participants, 13 of them self-reported that they would like to use DeepSeer when developing and debugging RNN models in the future, while 1 participant stayed neutral. The median is 6.5 on a 7-point Likert scale (1—I don't want to use it at all, 7—I will definitely use it if available). We report participants' qualitative feedback on the key features of DeepSeer from both post-study survey and user study recordings below.

Intermediate Prediction Results. All 14 participants using DeepSeer found the on-demand intermediate prediction results provided by DeepSeer useful. The median rating is 7 out of 7. P9 mentioned, "*stepping through intermediate predictions help me understand why model makes a wrong prediction. For example, the text is apparently about sports. However, the model goes into state 20 which is not quite related to sports.*" Moreover, participants also liked the color-coding of intermediate prediction results.

State Diagram. Among 14 participants, 11 of them considered the state diagram in DeepSeer useful. The median rating is 6. P12 commented, "*extracting an RNN model as a state diagram is nice, and I think it will also be helpful when interpreting [RNN models] with more complex data such as medical data.*" While the majority of participants did not find the state diagram overwhelming, 3 found it slightly overwhelming and 1 found it very overwhelming. 7 out of 14 participants found it useful to interact with the state diagram, e.g., seeing statistical distribution over states and keywords associated with each state.

Pattern Summaries. 9 out of 14 participants found that seeing the patterns in the pattern summary view and filtering the dataset based on a specific pattern are useful (median rating: 6). P2 mentioned, "*it is good for us to see inside of the model and find the bug with possibly buggy patterns.*" In particular, participants also mentioned that the pattern summary view is helpful for debugging. P8 said, "*I can click the buggy patterns to check related sentences. This helps me identify why model usually mis-classify [sentences] with these patterns.*"

Searching and Filtering Instances. Most participants agreed that it is useful to interact with each data instance in the instance view (median rating: 7) and search for similar instances (median rating: 7). P4 mentioned, "*I like the colors associated with each label, I feel this helped a lot with looking at examples. I also liked how it clearly showed examples with their true class and prediction. Also, the ability to also filter examples by correctness, prediction, and the true label was very helpful for me.*"

Limitations and Suggestions. 5 out of 14 participants pointed out that it would be better if DeepSeer could provide additional statistical information about model accuracy, e.g., confusion matrix. 1 participant suggested that adding the confidence score for the explanation could help them make more informed decisions. 1 participant found the state diagram mentally demanding. P6 said, "*state diagram seemed a bit hard to interpret by just looking at it and*

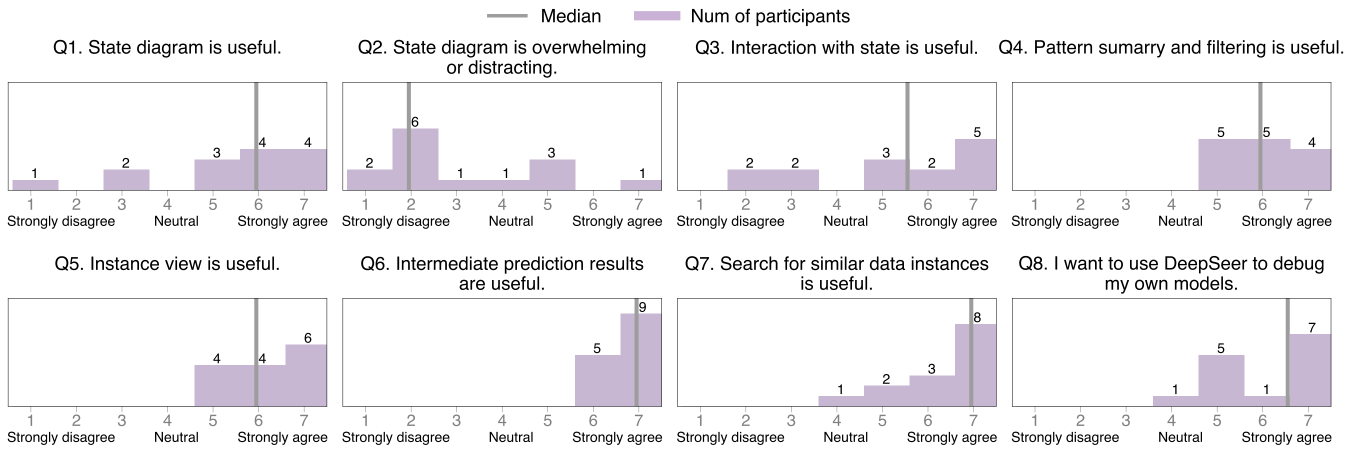


Figure 10: Participants' ratings about DeepSeer's tool features (1 is "strongly disagree" and 7 is "strongly agree")

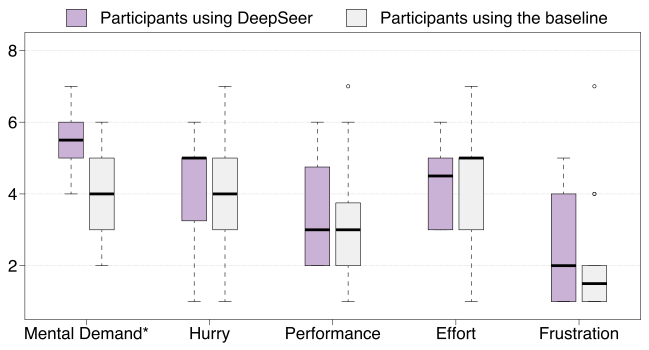


Figure 11: Participants' ratings about cognitive load (1 means "strongly disagree" and 7 means"strongly agree", * means the mean difference is statistically significant.)

probably wouldn't be immediately intuitive to a user opening this application up initially."

Cognitive Overhead. In the post-study survey, participants rated the cognitive load of the study via the NASA TLX questionnaire [21]. Fig. 11 shows their ratings for the five NASA TLX questions. We found that there was no significant difference when using DeepSeer vs. LIME in terms of hurry, performance, effort, and frustration (Welch's t-test: $p = 0.7731$, $p = 0.7244$, $p = 0.6916$, and $p = 0.5620$). Since DeepSeer renders much more information about model behavior (e.g., a state diagram, a pattern view, on-demand intermediate prediction results), participants using DeepSeer felt more mental demand (median value: 5 vs. 4, Welch's t-test: $p = 0.0011$).

9 DISCUSSION

9.1 Design Implications

The user study results suggest that DeepSeer helps users achieve a more comprehensive understanding of the assigned model, and perform better on model debugging compared with the baseline tool, LIME [53]. We believe this is largely attributed to DeepSeer's interactive support for explaining the model's global and local behavior. While a few studies have discussed about the importance of global and local explanations [24, 45], our work provides specific insights on how to support global and local explanations in a unified interface for RNN models.

In DeepSeer, global explanations are mainly rendered in the *State Diagram View* and the *Pattern Summary View*. The abstracted state diagram helps users interpret the hidden states and complex transitions among these states, while the summarized text patterns help users quickly identify either influential or buggy patterns learned by the model. These global explanations boost users' understanding and debugging process. Despite all the benefits of global explanations, we found it still necessary for participants to have the instance-level explanation to contextualize their understanding of model behavior. In particular, given a specific state or text pattern, user study participants often got curious about how it sounds in different texts. In the post-study survey, they highly appreciated the *Intermediate Prediction Results* feature. DeepSeer allows users to zoom into local explanations by actively filtering instances based on selected states or patterns, as well as zooming back to the model's global behavior by tracing back to the state diagram. Through these ways, global and local explanations are served as a synergistic loop for model understanding and debugging.

Furthermore, we find that users cared about how the given explanations are derived from the internal decision-making process of an RNN model. When using LIME [53], 4 out of 14 participants using LIME questioned the explanations (highlighted keywords) given by LIME. For instance, P24 commented in the post-study survey, *"I hope LIME can provide a reason why some words have a high sincere or insincere score."* As a more tangible and actionable solution, DeepSeer not only communicates the correlation between specific features in an input to a prediction result, but also communicates the internal decision-making process of a model. DeepSeer renders model's decision-making process in two ways. First, it renders the transition between different internal states of the model in the state diagram view. Second, for an individual prediction, it renders the intermediate prediction results as well as their correspondence to the internal states of a model. By inspecting such

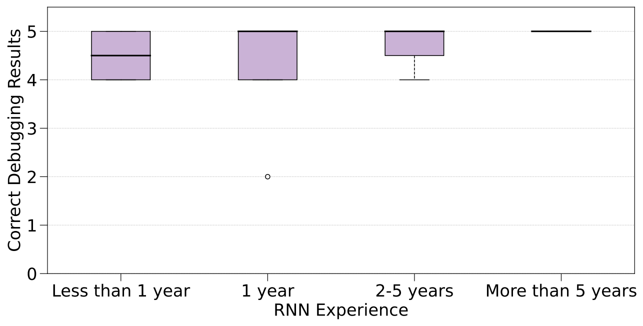


Figure 12: DeepSeer users' performance on model debugging with various levels of RNN expertise.

a decision-making process, users can better understand how the model arrives at a specific prediction and gain more trust from the generated explanations.

As an interactive XAI tool, it is also important to provide users with interpretable explanations, especially for RNN models. Note that a few prior techniques have tried to visualize the decision process of RNN [27, 59]. However, they usually only directly visualize the value of each hidden state. For instance, LSTMVis [59] visualizes the change of hidden state values in parallel coordinates. Given that hidden state values are essentially numerical values in a high-dimensional space, it is challenging to interpret their semantic meanings. To address this challenge, DeepSeer bundles hidden states with associated words and phrases in a text corpus and visualizes the transition between them as a state diagram. In this way, the internal decision-making process becomes more interpretable to non-experts.

9.2 Target Users and User Expertise

DeepSeer is designed for any developers who needs to train and debug an RNN model by themselves. They can be experienced ML developers, regular software developers who just started learning RNNs, or students who use RNN in a course project. In the user study, we recruited participants with diverse expertise in RNN, including 4 participants with less than 1 year of RNN experience, 5 with 1 year, 3 with 2—5 years, and 2 with more than 5 years. Our further analysis shows that, while participants with more RNN experience performed slightly better, the difference was not significant (Fig. 12). This implies the effectiveness of fDeepSeer is not strongly correlated to their expertise.

9.3 Generalization to Different ML Tasks and Models

Though our work has only evaluated DeepSeer on sentiment analysis and topic modeling tasks, we believe DeepSeer can generalize to different NLP Tasks as well. To reuse DeepSeer for other tasks, one may consider adapting the color mapping mechanism for abstract states. For example, for machine translation tasks, one can color each state according to the part-of-speech tag. By inspecting each state's color and associated words, users could interpret if an RNN model is translating a sentence correctly.

In this work, we focused on RNNs, which is a representative model architecture for processing sequential data. In addition to RNNs, it may be possible to use DeepSeer to interpret RNN variants such as Bidirectional-LSTM [37] or Transformers [60]. While the principle of Bidirectional-LSTM is similar to a naive RNN, some adaptions to the state abstraction method are required. For instance, one should consider collecting the model's hidden states when processing the input text in both two directions. Since transformers are permutation-invariant, they process all words in an input sentence at the same time, not sequentially. Therefore, we can no longer bundle a word with a hidden state. However, one can treat the output of each hidden layer as a concrete state. Then the transition can be built among different hidden layers instead of among different words.

9.4 Limitations and Future Work

One limitation of our user study design is that the comparison baseline, LIME [53], is designed for generating local explanations instead of global explanations. Thus, we cannot directly compare the global explanation effectiveness of DeepSeer to LIME. Besides, LIME is not specialized for RNN. While there are RNN-specific tools, such as LSTMVis [59] and RNNVis [43], we failed to run them on our RNN models after several attempts due to version compatibility issues. Both LSTMVis and RNNVis were built years ago on out-of-dated DL frameworks (TensorFlow r0.12 and Torch 7), which can no longer be used to analyze DL models built by later framework versions. Significant efforts are needed to re-implement them. Therefore, we consider such re-implementation out of the scope of this work. An alternative solution can be creating variants of DeepSeer by disabling some key features as baselines, which can help us better attribute the success of DeepSeer to individual features. This is worth investigating in future work.

As we are researchers from an R1 university, we do not have access to professional developers and data scientists who build RNN models in their work. Instead, we recruited graduate students who have experience in building RNN models. ML practitioners may share more interesting insights compared with graduate students.

Additionally, our user study has only evaluated DeepSeer on RNNs for sentiment analysis and topic classification. To comprehensively investigate the usefulness of DeepSeer, one can consider leveraging DeepSeer to understand and debug RNN models for other tasks, beyond text classification, e.g., machine translation.

Furthermore, our current design only supports visualizing and analyzing one RNN model. Once the bugs are identified with the help of DeepSeer, re-training the RNN model is needed. Therefore, a possible future direction is to develop tool support for comparing two or more versions of an RNN model [46]. Besides, one can also improve DeepSeer by designing tool support for model tracking [4] and model selection [10].

10 CONCLUSION

In this paper, we present a novel system called DeepSeer to help ML developers understand and debug recurrent neural networks. DeepSeer makes use of a state abstraction method that bundles semantically similar hidden states of an RNN model and abstracts it to a finite state machine. Through DeepSeer, users can explore both the model's global and local behavior, and also debug incorrect predictions. We demonstrate DeepSeer's usefulness and usability through a between-subjects user study with 28 model developers on two different RNN models. The results show that DeepSeer's tightly-coordinated views brought developers a deeper understanding of an RNN model compared with a popular XAI technique, LIME. Furthermore, participants using DeepSeer were able to identify the root causes of more incorrect predictions and provide more actionable plans to improve the RNN model. In the end, we discuss the design implications from DeepSeer, and propose several promising future work directions.

ACKNOWLEDGMENTS

We would like to thank all anonymous participants in the user study and anonymous reviewers for their valuable feedback. This work was supported in part by Amii RAP program, Canada CIFAR AI Chairs Program, the Natural Sciences and Engineering Research Council of Canada (NSERC No.RGPIN-2021-02549, No.RGPAS-2021-00034, No.DGECR-2021-00019), as well as JSPS KAKENHI Grant No.JP20H04168, JST-Mirai Program Grant No.JPMJMI20B8.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access* 6 (2018), 52138–52160.

[3] Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. 2020. Debugging Tests for Model Explanations. *Advances in Neural Information Processing Systems* 33 (2020), 700–712.

[4] Saleema Amershi, Max Chickering, Steven M Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 337–346.

[5] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-AI interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems*. 1–13.

[6] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115.

[7] Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. 2020. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1899–1909.

[8] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6541–6549.

[9] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181* (2017).

[10] Markus Bögl, Wolfgang Aigner, Peter Filzmoser, Tim Lammarsch, Silvia Miksch, and Alexander Rind. 2013. Visual analytics for model selection in time series analysis. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2237–2246.

[11] Carrie J Cai, Jonas Jongejan, and Jess Holbrook. 2019. The effects of example-based explanations in a machine learning interface. In *Proceedings of the 24th international conference on intelligent user interfaces*. 258–262.

[12] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. 103–111.

[13] Arun Das and Paul Rad. 2020. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371* (2020).

[14] Jonathan Dodge, Q Vera Liao, Yunfeng Zhang, Rachel KE Bellamy, and Casey Dugan. 2019. Explaining models: an empirical study of how explanations impact fairness judgment. In *Proceedings of the 24th international conference on intelligent user interfaces*. 275–285.

[15] Xiaoning Du, Yi Li, Xiaofei Xie, Lei Ma, Yang Liu, and Jianjun Zhao. 2020. Marble: Model-based robustness analysis of stateful deep learning systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 423–435.

[16] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. 2019. Deepstellar: Model-based quantitative analysis of stateful deep learning systems. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 477–487.

[17] Mary T Dzindolet, Scott A Peterson, Regina A Pomranky, Linda G Pierce, and Hall P Beck. 2003. The role of trust in automation reliance. *International journal of human-computer studies* 58, 6 (2003), 697–718.

[18] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14, 2 (1990), 179–211.

[19] Philippe Fournier-Viger, Antonio Gomariz, Ted Gueniche, Espérance Mwamikazi, and Rincy Thomas. 2013. TKS: efficient mining of top-k sequential patterns. In *International Conference on Advanced Data Mining and Applications*. Springer, 109–120.

[20] Antonio Gulli. 2005. AG's corpus of news articles. http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

[21] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.

[22] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. 241–250.

[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[24] Fred Hohman, Andrew Head, Rich Caruana, Robert DeLine, and Steven M Drucker. 2019. Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–13.

[25] Zhihua Jin, Yong Wang, Qianwen Wang, Yao Ming, Tengfei Ma, and Huamin Qu. 2022. Gnnlens: A visual analytics approach for prediction error diagnosis of graph neural networks. *IEEE Transactions on Visualization and Computer Graphics* (2022).

[26] Minsuk Kahng, Pierre Y Andrews, Aditya Kalro, and Duen Horng Chau. 2017. Activis: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 88–97.

[27] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015).

[28] Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, and Jennifer Wortman Vaughan. 2020. Interpreting Interpretability: Understanding Data Scientists' Use of Interpretability Tools for Machine Learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[29] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*. PMLR, 2668–2677.

[30] René F Kizilcec. 2016. How much information? Effects of transparency on trust in an algorithmic interface. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2390–2395.

[31] Pang Wei Koh, Kai-Siang Ang, Hubert H. K. Teo, and Percy Liang. 2019. On the Accuracy of Influence Functions for Measuring Group Effects. In *Advances in Neural Information Processing Systems 32 (2019)*. 5255–5265.

[32] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*. PMLR, 1885–1894.

[33] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and Understanding Neural Models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 681–691.

[34] Rongjian Li, Wenlu Zhang, Heung-Il Suk, Li Wang, Jiang Li, Dinggang Shen, and Shuiwang Ji. 2014. Deep learning based imaging data completion for improved brain disease diagnosis. In *International conference on medical image computing and computer-assisted intervention*. Springer, 305–312.

[35] Q Vera Liao, Daniel Gruen, and Sarah Miller. 2020. Questioning the AI: informing design practices for explainable AI user experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–15.

[36] Zachary C Lipton. 2018. The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue* 16, 3 (2018), 31–57.

[37] Gang Liu and Jiabao Guo. 2019. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing* 337 (2019), 325–338.

[38] Shusen Liu, Zhimin Li, Tao Li, Vivek Srikumar, Valerio Pascucci, and Peer-Timo Bremer. 2018. Nlize: A perturbation-driven visual interrogation tool for analyzing and interpreting natural language inference models. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 651–660.

[39] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[40] Shiqing Ma, Yousra Aafer, Zhaogui Xu, Wen-Chuan Lee, Juan Zhai, Yingqi Liu, and Xiangyu Zhang. 2017. LAMP: data provenance for graph based machine learning algorithms through derivative computation. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 786–797.

[41] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama. 2018. MODE: automated neural network model debugging via state differential analysis and input selection. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 175–186.

[42] Geoffrey J McLachlan and Kaye E Basford. 1988. *Mixture models: Inference and applications to clustering*. Vol. 38. M. Dekker New York.

[43] Yao Ming, Shaozu Cao, Ruixiang Zhang, Zhen Li, Yuanzhe Chen, Yangqiu Song, and Huamin Qu. 2017. Understanding hidden memories of recurrent neural networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 13–24.

[44] Christoph Molnar. 2020. *Interpretable machine learning*. Lulu. com.

[45] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* 73 (2018), 1–15.

[46] Sugeerth Murugesan, Sana Malik, Fan Du, Eunyee Koh, and Tuan Manh Lai. 2019. Deepcompare: Visual and interactive comparison of deep learning model performance. *IEEE computer graphics and applications* 39, 5 (2019), 47–59.

[47] Takamasa Okudono, Masaki Waga, Taro Sekiyama, and Ichiro Hasuo. 2020. Weighted automata extraction from recurrent neural networks via regression on state spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5306–5314.

[48] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. 2018. The building blocks of interpretability. *Distill* 3, 3 (2018), e10.

[49] Zi Peng, Jinqiu Yang, Tse-Hsun Chen, and Lei Ma. 2020. A first look at the integration of machine learning models in complex autonomous driving systems: A case study on apollo. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1240–1250.

[50] Quora. 2019. Quora insincere questions classification. https://www.kaggle.com/c/quora-insincere-questions-classification/data

[51] Guillaume Rabusseau, Tianyu Li, and Doina Precup. 2019. Connecting weighted automata and recurrent neural networks through spectral learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1630–1639.

[52] Amir Hossein Akhavan Rahnama and Henrik Boström. 2019. A study of data and label shift in the LIME framework. *arXiv preprint arXiv:1910.14421* (2019).

[53] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[54] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[55] Tobias Schneider, Joana Hois, Alischa Rosenstein, Sabiha Ghellal, Dimitra Theofanou-Fülbier, and Ansgar RS Gerlicher. 2021. ExplAIn Yourself! Transparency for Positive UX in Autonomous Driving. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[56] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Pariah, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[57] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

[58] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017).

[59] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. 2017. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 667–676.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[61] Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. Extracting automata from recurrent neural networks using queries and counterexamples. In *International Conference on Machine Learning*. PMLR, 5247–5256.

[62] Xiaofei Xie, Wenbo Guo, Lei Ma, Wei Le, Jian Wang, Lingjun Zhou, Yang Liu, and Xinyu Xing. 2021. RNNrepair: Automatic RNN repair via model-based analysis. In *International Conference on Machine Learning*. PMLR, 11383–11392.

[63] Bing Yu, Hua Qi, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, and Jianjun Zhao. 2022. DeepRepair: Style-Guided Repairing for Deep Neural Networks in the Real-World Operational Environment. *IEEE Transactions on Reliability* 71, 4 (2022), 1401–1416. https://doi.org/10.1109/TR.2021.3096332

[64] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.

[65] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems* 28 (2015).

A STATE ABSTRACTION

In this section, we present the technical details of state abstraction for an RNN model. The core idea is to extract all possible hidden states of an RNN model using training data and then group similar hidden states to build a finite state machine (FSM). Algorithm A1 shows the procedure of state abstraction of an RNN model.

Algorithm A1: State abstraction of an RNN model.

```
Input: a trained RNN model $R$, training data $\mathcal{X}_{train}$, PCA
       dimension $k$, number of states $n$
Output: PCA model $P$, GMM model $G$, an abstraction
        model $A = \{P, G\}$
1 $\mathcal{H} \leftarrow \{\}$;
2 for $x$ in $\mathcal{X}_{train}$ do
3     $H \leftarrow$ record_hidden_states$(R, x)$;
4     $\mathcal{H}$.append$(H)$;
5 end
6 $P$ = PCA$(\mathcal{H}, k)$;
7 $G$ = GMM$(P(\mathcal{H}), n)$;
8 return $A$;
```

Algorithm A1 takes two inputs: a trained RNN model R and training data X_{train}, as well as two parameters: PCA (principal component analysis) dimension p and number of abstracted states n. Given a trained RNN model, we first iterate through all the training data $\mathcal{X}_{train}$ to collect all possible hidden states $\mathcal{H}$ from the RNN model (Line 1:5). Line 3 records all the hidden states H in an RNN model when processing a specific input instance x. Suppose an input instance x has l words, then l different hidden states will be produced when the RNN model processes these l words sequentially. For example, given the sentence "I love Machine Learning", the RNN model will process four words: "I", "love", "machine", and "learning" sequentially. Therefore, four different hidden state vectors will be produced and recorded when the RNN model processes this sentence.

After recording all the hidden states using the training data, we create a PCA model k to reduce the dimension of these hidden states into p (Line 6). Meanwhile, we also obtain a PCA model P. Now we abstract $|P(\mathcal{H})|$ dimension reduced hidden states into n abstracted states. Different from the prior work [16], which uses a grid-based method, we adopt a GMM (Gaussian mixture model [42]) G to cluster these dimension-reduced hidden states (Line 7). After executing Line 7, we obtain a GMM model G. Our abstracted model As has now been built, which consists of a PCA model P and a GMM model G. Note that both PCA and GMM are implemented with scikit-learn with default parameters except "n_components".

In our usage scenario and user study, we set the PCA dimension k as 20 and the number of states n as 40. We further show that the abstraction model using this setting can provide consistent predictions compared with the original RNN model in Appendix B.

B FAITHFULNESS OF STATE ABSTRACTION

In this section, we show that the abstracted model (i.e., the finite state machine) can make consistent predictions as the original RNN model in three different tasks, one from the usage scenario (Section 6) and the other two from the user study (Section 7.2). We measure the prediction consistency between the finite state machine and the original RNN. Suppose the dataset is $X = \{x_1, x_2, \ldots, x_N\}$, the abstracted model is $\mathcal{F}$, and the RNN model is $\mathcal{R}$. The prediction consistency can be calculated through Eq. 2.

$$prediction\_consistency = \frac{\sum_{i=1}^{N} \mathcal{F}(x_i) == R(x_i)}{N} \quad (2)$$

Table B1 shows the prediction consistency of the two models in each task on the training and test data separately. We can see that for the binary classification models (Toxic and Quora), the abstraction models can provide highly consistent predictions (consistency is 99% and 97%) compared with the original RNN models on both training and test data. For the multi-classification model (AGNews), the abstraction model can still provide very consistent predictions (consistency is 86%). These results demonstrate the faithfulness of our state abstraction technique.

C NUMBER OF ABSTRACTED STATES

During our user study, we set the number of abstracted states to 40. This number is empirically decided to achieve a good balance between the prediction consistency to the original RNN and the cognitive effort of inspecting a state diagram. To further show that this setting will not affect the abstraction model's faithfulness, we report the abstracted models' prediction consistency w.r.t. the number of states of all three models in Fig. C1.

As we can see, a lower number of states will lead to a lower prediction consistency. However, when the number of states is larger than 40, the prediction consistency stays largely the same. Therefore, we choose this number of states, 40, throughout our motivating example and user study.

D ML TASKS

D.1 ML Task 1: Sentimental Analysis (Quora dataset)

Task Description:

In this task, participants were given an RNN model trained on the Quora dataset.

Quora dataset is collected from quora.com, where each text in the dataset is labeled as "Sincere" or "Insincere". An insincere question is defined as a question intended to make a statement rather than look for helpful answers.

Participants first used the tool to understand the model. They were asked to use the tool to explore the model's behaviours and performance on training data and test data. After exploring, participants were asked to share their findings, e.g., Did they find any insights? Did they find any bugs? How would they improve this model?

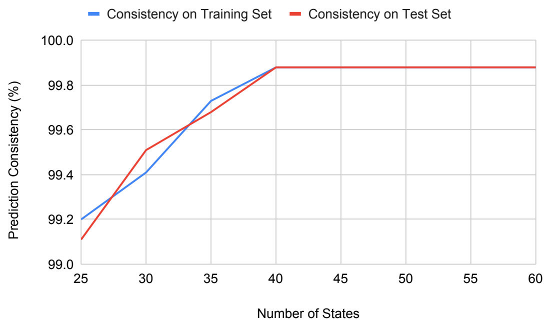
Then participants were given 5 misclassified sentences. Participants had 10 minutes in total to finish the following task: for each sentence, participants needed to find out why this sentence is misclassified with the help of DeepSeer.

D.2 ML Task 2: Topic Classification (AGNews dataset)

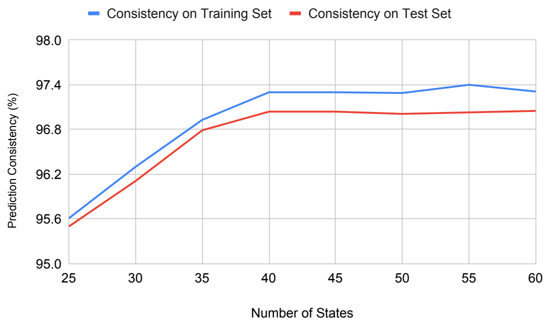
Task Description:

Table B1: Quantitative comparison between the abstraction model predictions to the original RNN's predictions.

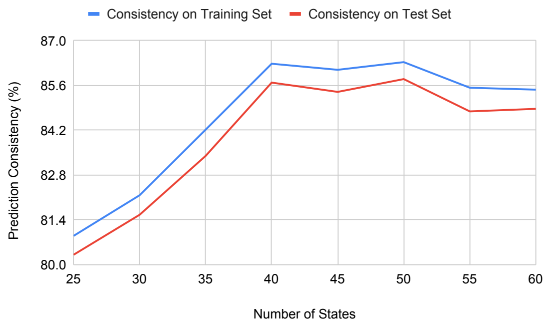
	Prediction consistency on training set	Prediction consistency on test set
Toxic	99.88%	99.88%
Quora	97.30%	97.04%
AGNews	86.28%	85.59%



(a) Prediction consistency between the original RNN model and the abstracted model on Toxic dataset.



(b) Prediction consistency between the original RNN model and the abstracted model on Quora dataset.



(c) Prediction consistency between the original RNN model and the abstracted model on AGNews dataset.

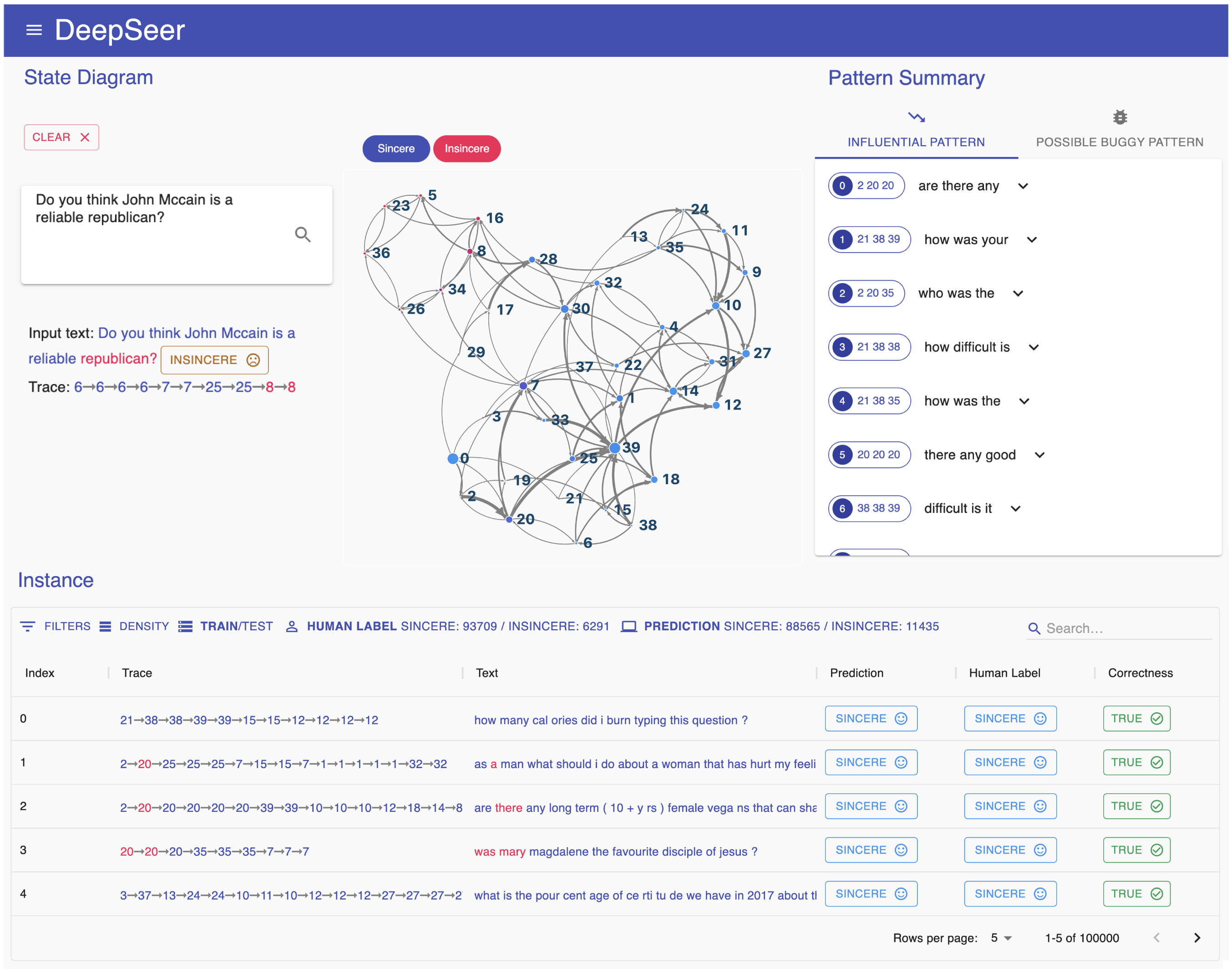
Figure C1: Prediction consistency w.r.t. the number of states of all three RNN models' abstracted models.

In this task, participants were given an RNN model trained on the AGNews dataset.

AGNews dataset is a collection of news articles. This RNN model classifies each text into different topics, including World, Sports, Business, and Sci/Tech.

Participants first used the tool to understand the model. They were asked to use the tool to explore the model's behaviours, and performance on training data and test data. After exploring, participants were asked to share their findings, e.g., Did they find any insights? Did they find any bugs? How would they improve this model?

Then participants were given 5 misclassified sentences. Participants had 10 minutes in total to finish the following task: for each sentence, participants needed to find out why this sentence is misclassified with the help of DeepSeer.


DeepSeer

State Diagram

CLEAR ✕

Do you think John Mccain is a reliable republican?

Input text: Do you think John Mccain is a reliable republican? INSINCERE

Trace: 6→6→6→6→7→7→25→25→8→8

Pattern Summary

INFLUENTIAL PATTERN POSSIBLE BUGGY PATTERN

0	2 20 20	are there any
1	21 38 39	how was your
2	2 20 35	who was the
3	21 38 38	how difficult is
4	21 38 35	how was the
5	20 20 20	there any good
6	38 38 39	difficult is it

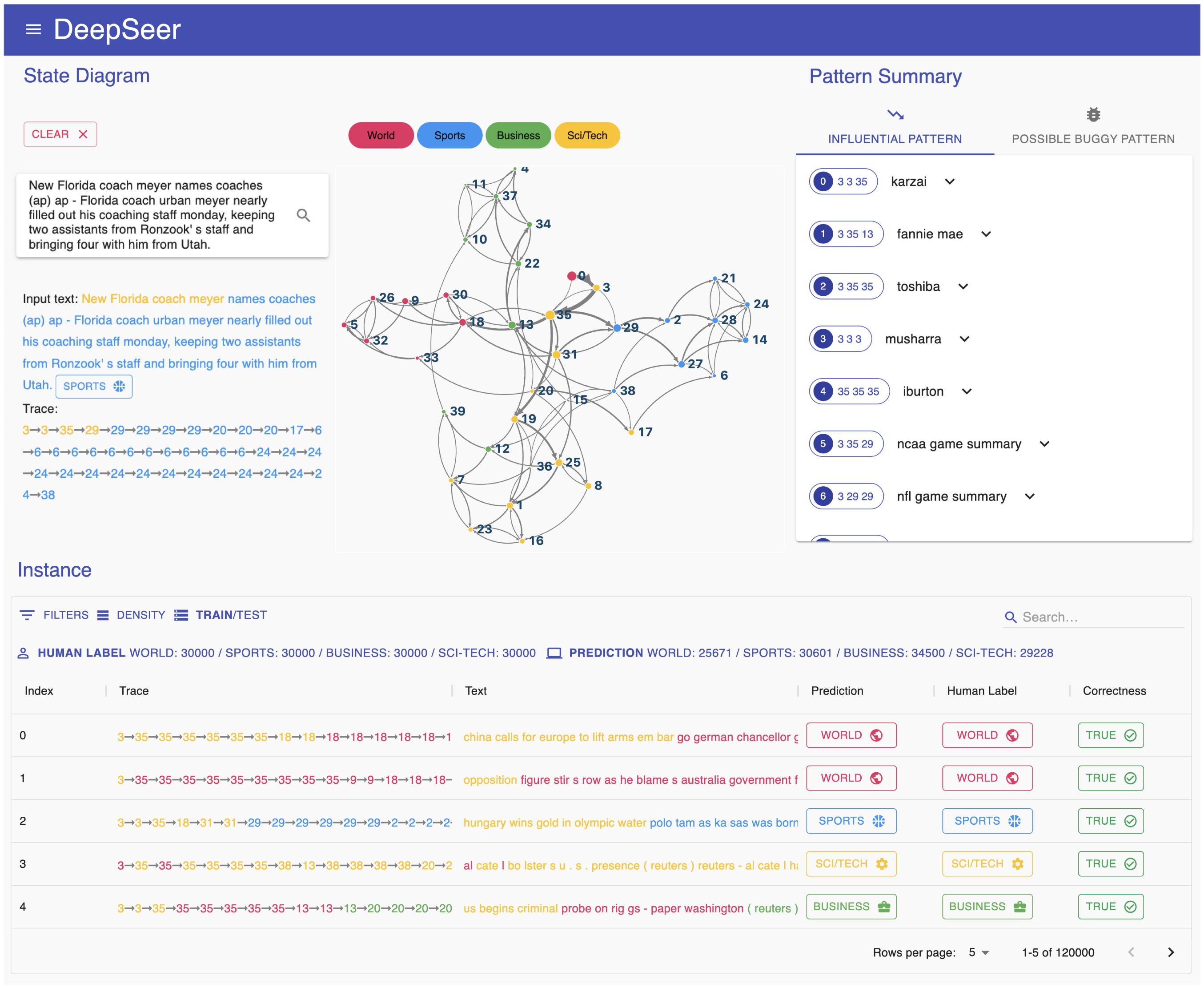
Instance

FILTERS DENSITY TRAIN/TEST HUMAN LABEL SINCERE: 93709 / INSINCERE: 6291 PREDICTION SINCERE: 88565 / INSINCERE: 11435 Search…

Index	Trace	Text	Prediction	Human Label	Correctness
0	21→38→38→39→39→15→15→12→12→12→12	how many cal ories did i burn typing this question ?	SINCERE	SINCERE	TRUE
1	2→20→25→25→25→7→15→15→7→1→1→1→1→1→32→32	as a man what should i do about a woman that has hurt my feeli	SINCERE	SINCERE	TRUE
2	2→20→20→20→20→20→39→39→10→10→10→12→18→14→8	are there any long term (10 + y rs) female vega ns that can sha	SINCERE	SINCERE	TRUE
3	20→20→20→35→35→35→7→7→7	was mary magdalene the favourite disciple of jesus ?	SINCERE	SINCERE	TRUE
4	3→37→13→24→24→10→11→10→12→12→12→27→27→27→2	what is the pour cent age of ce rti tu de we have in 2017 about tl	SINCERE	SINCERE	TRUE

Rows per page: 5 1-5 of 100000

Figure D2: The interface of DeepSeer used for ML task 1 (Quora dataset).



New Florida coach meyer names coaches (ap) ap - Florida coach urban meyer nearly filled out his coaching staff monday, keeping two assistants from Ronzook' s staff and bringing four with him from Utah.

Input text: New Florida coach meyer names coaches (ap) ap - Florida coach urban meyer nearly filled out his coaching staff monday, keeping two assistants from Ronzook' s staff and bringing four with him from Utah. SPORTS

Trace:
3→3→35→29→29→29→29→29→20→20→20→17→6→6→6→6→6→6→6→6→6→6→6→6→6→24→24→24→24→24→24→24→24→24→24→24→24→24→24→24→38

0	3 3 35	karzai
1	3 35 13	fannie mae
2	3 35 35	toshiba
3	3 3 3	musharra
4	35 35 35	iburton
5	3 35 29	ncaa game summary
6	3 29 29	nfl game summary

HUMAN LABEL WORLD: 30000 / SPORTS: 30000 / BUSINESS: 30000 / SCI-TECH: 30000 PREDICTION WORLD: 25671 / SPORTS: 30601 / BUSINESS: 34500 / SCI-TECH: 29228

Index	Trace	Text	Prediction	Human Label	Correctness
0	3→35→35→35→35→35→35→18→18→18→18→18→18→18→1	china calls for europe to lift arms em bar go german chancellor g	WORLD	WORLD	TRUE
1	3→35→35→35→35→35→35→35→35→35→9→9→18→18→18–	opposition figure stir s row as he blame s australia government f	WORLD	WORLD	TRUE
2	3→3→35→18→31→31→29→29→29→29→29→29→2→2→2→2·	hungary wins gold in olympic water polo tam as ka sas was born	SPORTS	SPORTS	TRUE
3	3→35→35→35→35→35→35→38→13→38→38→38→38→20→2	al cate l bo lster s u . s . presence (reuters) reuters - al cate l hi	SCI/TECH	SCI/TECH	TRUE
4	3→3→35→35→35→35→35→35→13→13→13→20→20→20→20	us begins criminal probe on rig gs - paper washington (reuters)	BUSINESS	BUSINESS	TRUE

Rows per page: 5 1-5 of 120000

Figure D3: The interface of DeepSeer used for ML task 2 (AGNews dataset).