DeepState: Selecting Test Suites to Enhance the Robustness of Recurrent Neural Networks

Authors: Zixi Liu,Yang Feng, Yining Yin, Zhenyu Chen

The Problem Addressed

Challenges associated with traditional RNN testing methods:

- **High Time and Resource Consumption:** Large datasets require significant time and resources for testing.
- **Inefficient Test Case Selection:** Existing methods might miss critical or diverse test cases, leading to insufficient coverage.

Motivation

- Recurrent Neural Networks (RNNs) are becoming increasingly prevalent in various critical applications, including areas like natural language processing, speech recognition, and machine translation.

- There is a need for more efficient and effective testing methodologies ensuring the robustness and reliability

Proposed Solution

- DeepState, which is a test suite selection tool for FNN models to reduce data labeling and computational cost.
- Efficiently selects a smaller, representative subset of test cases from the original dataset.
- Ranks the selected test cases based on their potential to reveal errors and weaknesses in the RNN model.

Proposed Solution

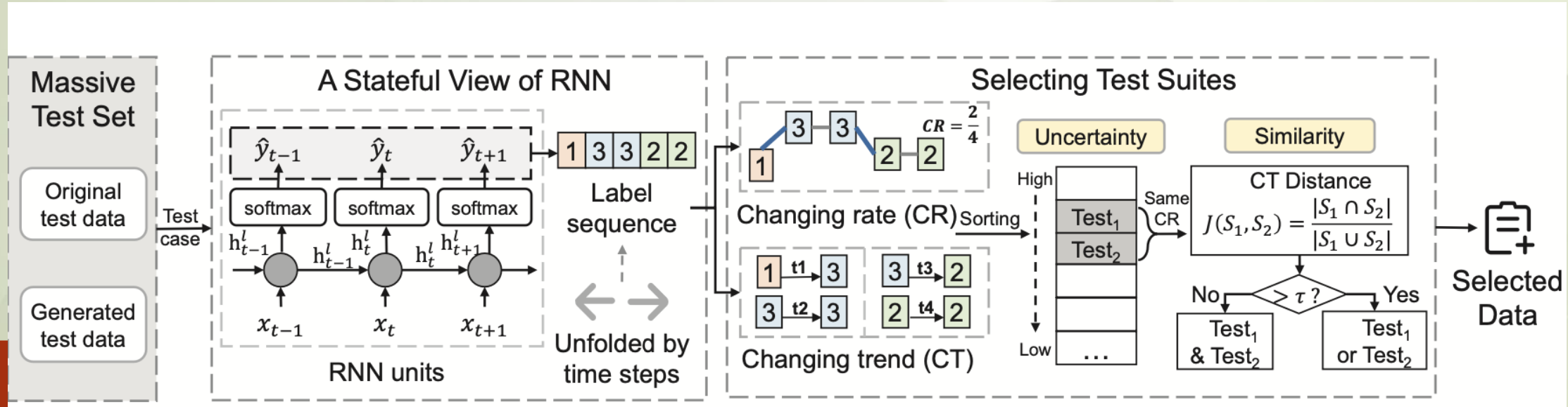

Figure 2: Overview of DeepState.

Research Questions Addressed

RQ1: Can the dataset selected by DeepState detect more defects than neuron-coverage-based methods?

RQ2: Can DeepState filter out the bug test cases in the candidate test data set?

RQ3: Can DeepState guide the retraining of an RNN to improve its accuracy?

Evaluation

- Adaptation of Inclusiveness
- Effectiveness in Bug Detection
- Inclusiveness Evaluation

Table 4: The bug detection rate of 10% and 20% selected tests.

	Model	Sel	Ran.	HSCov (CAM)	SC (CAM)	SC (CTM)	NC (CAM)	NC (CTM)	BSCov (CTM)	BTCov (CTM)	Deep State
MNIST	LSTM	10%	0.33	0.55	0.32	0.18	0.33	0.1	0.14	0.23	**0.63**
		20%	0.32	0.44	0.33	0.18	0.33	0.09	0.13	0.19	**0.58**
	BiLSTM	10%	0.33	0.61	0.33	0.32	0.39	0.23	0.08	0.08	**0.69**
		20%	0.33	0.49	0.33	0.32	0.36	0.23	0.1	0.1	**0.63**
Fashion	LSTM	10%	0.32	0.46	0.32	0.21	0.32	0.22	0.21	0.25	**0.6**
		20%	0.32	0.39	0.32	0.22	0.33	0.24	0.22	0.26	**0.55**
	GRU	10%	0.32	0.42	0.32	0.31	0.32	0.32	0.3	0.29	**0.61**
		20%	0.32	0.37	0.32	0.31	0.32	0.3	0.3	0.24	**0.56**
Snips	BiLSTM	10%	0.12	0.12	0.13	0.11	0.13	0.09	0.1	0.14	**0.31**
		20%	0.12	0.12	0.12	0.12	0.13	0.1	0.11	0.13	**0.27**
	GRU	10%	0.1	0.11	0.1	0.11	0.1	0.12	0.03	0.05	**0.3**
		20%	0.1	0.11	0.11	0.11	0.11	0.12	0.07	0.08	**0.25**
AgNews	LSTM	10%	0.18	0.24	0.18	0.18	0.18	0.15	0.13	0.14	**0.48**
		20%	0.18	0.22	0.18	0.18	0.19	0.16	0.12	0.13	**0.42**
	BiLSTM	10%	0.21	0.26	0.2	0.2	0.2	0.17	0.12	0.14	**0.46**
		20%	0.2	0.25	0.2	0.2	0.2	0.18	0.13	0.14	**0.41**

Results:

- Evaluation results on code change quality estimation , review generation and code refinement tasks

Novelty Of The Solution

- **Incorporation of Both CTM and CAM**: DeepState combines the advantages of both CTM (Changing Rate-based Test Case Selection) and CAM (Changing Trend-based Test Case Selection) methods. This allows for the selection of test cases based on changing rate and similarity of changing trends, resulting in a more diverse and effective test set.

- **Improved State Transition Capture**: DeepState's hidden neuron output value-based selection strategy facilitates better capture of the state transition of RNN based on time steps, enhancing its performance in detecting potential defects under resource constraints.

Novelty Of The Solution

- **Diversity in Test Case Selection**: The selection strategy of DeepState results in a more diverse test set, which is conjectured to fundamentally benefit its performance in bug detection.

- **Effectiveness in Bug Detection**: DeepState has demonstrated higher bug detection rates compared to random selection strategies and other existing methods (such as HSCOV, SC, and NC), indicating its effectiveness in helping RNN models detect potential defects within given resource constraints.

Assumptions

- **Test Subject Selection**

  Four commonly used datasets are used and two RNN models .

- **Availability of Labeled Data**

  The simulation assumes the availability of labeled data for training and evaluating the model.

- **Homogeneity of Data**

  The simulation might assume homogeneity in the data, implying that all data points are drawn from the same distribution or exhibit similar characteristics

Limitations

- **Generalizability**:
  The tool they made might not work as well for every single kind of RNN or every task you want to use an RNN for, just like a toy that's fun in the playground but not as much fun at home.

- **Dependency on Dataset and Model Selection**:
  They chose specific RNN models and types of data to test their tool, which is like picking only certain flavors of ice cream to taste; it doesn't tell you how good the tool is with flavors they didn't try.

- **Simulation of Test Data:**
  The tests they made for the RNNs are like pretend play scenarios; they're not sure if these pretend plays will help the RNN perform well in real life, with real problems and real data.

- **Parameter Settings:**
  They set some dials and knobs in a certain way when using their tool because they think it's the best way, but it's like setting up a video game to easy mode and not knowing if it will be too hard or still work on the harder levels.

Practical significance

- **Saves Time and Resources**
  DeepState helps to pick the most useful tests from a big pile of possible tests, which means you can spend less time and use fewer resources to make your RNN smarter.

- **Finds Mistakes Better**
  It's like having a super magnifying glass that finds where the smart system might trip up, better than other tools.

- **Works with Different RNNs**
  Whether your RNN is a small toy robot or a big factory machine, DeepState is designed to work with different kinds and sizes of RNNs, making it a handy tool in many places.

- **No Need for Tons of Data**
  Instead of feeding your RNN all the data you can find, DeepState figures out which pieces of data are like the 'nutritious' parts that help it learn the best.

- **Practical for Real-world Use**
  It's not just a cool idea for a science fair; DeepState is made to be used in the real world, helping real smart systems avoid real mistakes.

Discussion Point:1

- DeepState, currently applied to enhance the robustness of RNNs, can it be adapted to improve the robustness and efficiency of other deep learning models like CNNs or Transformer-based models across various applications?

Discussion Point:2

- The paper discusses augmentation techniques applied to MNIST, Fashion, Snips, and AgNews datasets but lacks clarity on the specific variations or methods employed during the augmentation process?

Discussion Point:3

- DeepState uses less complex datasets like MNIST to evaluate its performance. What happens when it is tested on more complex real-world datasets?

Thank You