Simulated Selfhost Bug Reproducibility

Mengying Bi, Qiaoyu Jia, Win Nawat Suvansinpan, Yiming Liu
Master of Science in Data Science, University of Washington

Project Motivation

Microsoft's Simulated Selfhost automatically generates and executes random action sequences in Windows to test product reliability. Rich information is embedded in the data, but it is not easy to extract.

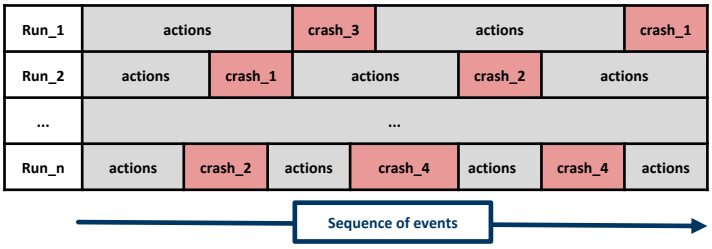
This project aims to:

- Provide engineering team with the smallest set of actions and their reproducibility estimate to recreate crashes and fix the underlying bugs
- Provide insights on the effectiveness of various methodologies

The project will lay the groundwork for future work involving Simulated Selfhost.

Dataset

- **Event Sequences**: Simulated Selfhost runs with action hashes and crash IDs.



- **Action Details:** Action hashes and their corresponding operations performed by the automated agent (ex. left click, tab, start menu, open settings)
- **Metadata**: Run metadata such as start/end time and crash details

Metrics

We define 2 metrics to measure the model performance throughout our project.

1. Offline Reproducibility Estimate*

A measurement applied to evaluate the likelihood that an action set would cause a crash.

For an action set *X* = [*A1, A2, …, An*] with crash *R*,

$$\text{Reproducibility} = P(R|X) = \frac{P(X \cap R)}{P(X)} \approx \frac{\#(Runs\ with\ X\ found\ preceding\ R)}{\#(Runs\ with\ X\ found)}$$

Hence, we want to find the set of action candidates with the highest reproducibility estimate.

** Offline estimate is calculated from the dataset instead of actually executing the action sequence on the testing agent.*

2. Capture Rate

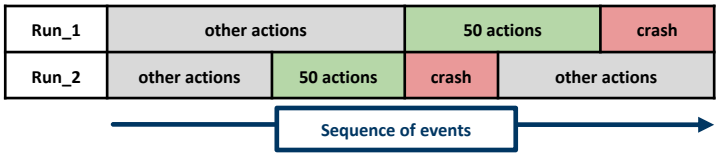
An action set with a high reproducibility estimate may only capture a small proportion of crash occurrences. We want to consider the number of times the action set is observed, given the occurrence of crash R.

$$\textit{Capture Rate} = P(X|R) \approx \frac{\#(Runs\ with\ X\ found\ preceding\ R)}{\#(Runs\ with\ R\ found)}$$

Approach

Hypothesis

n consecutive actions performed before a crash, produce the crash. For this project, *n* is set at 50.



Training Data for Classifier Models

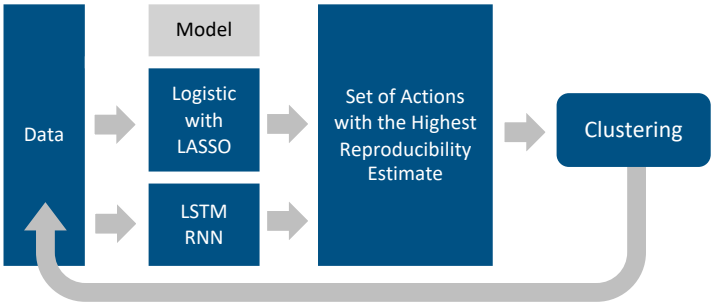
Positive Sample

- 50 actions preceding the crash from runs containing the crash

Negative Sample

- Find the most common actions among positive samples
- Pick windows of 50 actions containing common actions from runs with no crashes

label	actions
Positive Samples	
crash	50 actions
...	...
crash	50 actions
Negative Samples	
no crash	50 actions
...	...
no crash	50 actions



Workflow Schematic and Feedback Loop

- Train Logistic Regression with LASSO and LSTM models on data to find a set of actions with the highest reproducibility estimate
- Produce subsets of similar actions sequences by applying cluster analysis on action sequences from crashes that have high reproducibility estimate
- Train models on each data subset to produce action sets with higher reproducibility estimate

Models

Logistic Regression with L1 Regularization (LASSO)

- **Assumption**: Setup actions that cause a crash are non-sequential
- Train a logistic regression binary classifier for every crash (crash vs no crash) using preceding actions as features
- Select *n* features with the highest coefficient and check the reproducibility estimate of combinations of top features
- From our attempts on different *n* values, we do not see notable improvements beyond n=7

Long Short-Term Memory Recurrent Neural Network

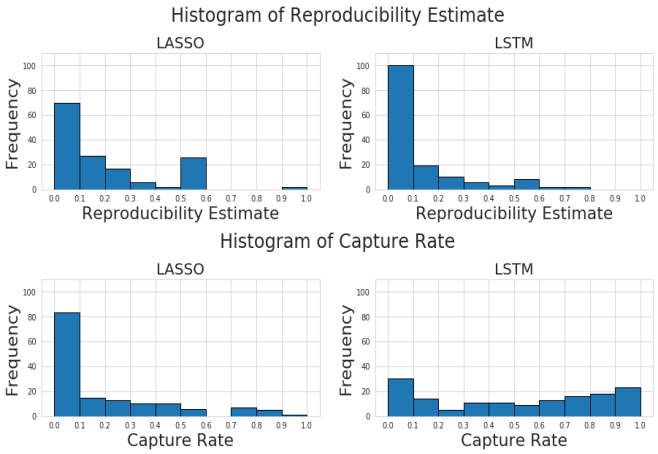
- **Assumption:** Actions that cause a crash are sequential
- Train an LSTM binary classifier for every crash (crash vs no crash) using preceding actions as features
- Find an action sequence with highest model score and check reproducibility estimate of combinations of actions in the sequence

Results

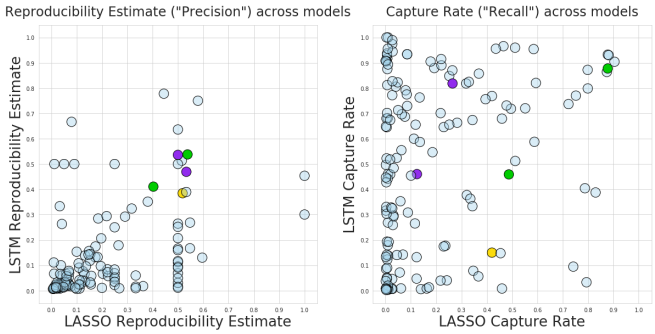
Deploying the models on 150 crashes that occur in 100 or more runs.

LASSO outperforms LSTM in **117** crashes
They have the same score in **5** crashes
LSTM outperforms LASSO in **28** crashes

	LASSO	LSTM
Average Reproducibility Estimate	0.1914	0.125
Number of action sets with Reproducibility Estimate > 40%	30	15
Average Capture Rate	0.1943	0.5048



LASSO returns action sets with higher reproducibility estimates, but LSTM's action sets have higher capture rates in general.

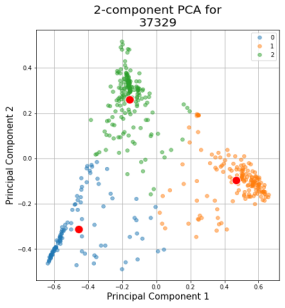


High reproducibility est. and capture rate by LSTM

High reproducibility est. and capture rate by LASSO

High reproducibility est. and capture rate by both models

The scatterplots show each model's performance. Colored circles represent crashes we warrant further investigation. Clustering Analysis shows that for some crashes, there are multiple action sets that can reproduce a crash.



For instance, there are 3 ways to reproduce crash 37329. Retraining models with subsets found by clustering action sequences preceding this crash will result in a 2% increase in the reproducibility estimate.

Conclusion

- Both LASSO and LSTM can uncover action sets with high reproducibility estimates (>40%)
- Characteristics of setup actions vary greatly
- LASSO is better at finding actions sets with high reproducibility estimates, but LSTM is better at finding actions sets with high capture rates
- Training models on sub-clusters can improve reproducibility estimates

Future Work

- Measure reproducibility by executing the set of action candidates on Simulated Selfhost agent
- Investigate how different window sizes affect performance. The current window size is 50 actions.

Acknowledgements

We extend our sincere gratitude to the Microsoft Cosine Team for their meticulous guidance, especially our supervisor Mohammed Helal. We also thank UW Research Computing Lab for a much-needed assist with cloud computing.