USENIX Security '24 Artifact Appendix: VulSim: Leveraging Similarity of Multi-Dimensional Neighbor Embeddings for Vulnerability Detection

Samiha Shimmi
Northern Illinois University
sshimmi@niu.edu

Ashiqur Rahman
Northern Illinois University
ashiqur.r@niu.edu

Mohan Gadde
Northern Illinois University
mgadde1@niu.edu

Hamed Okhravi
MIT Lincoln Laboratory
hamed.okhravi@ll.mit.edu

Mona Rahimi
Northern Illinois University
mrahimi1@niu.edu

A Artifact Appendix

This section provides a clear description of the hardware, software, and configuration requirements of the paper mentioned in the title.

A.1 Abstract

Our artifact is the source code to reproduce the result of the paper. It contains the source code to run our model. We provided a full pre-processed dataset to reproduce the results of Table 3 of the paper which is our major claim. Our model relies on three pre-existing models namely Code2vec, SBERT and CodeBERT. To run our model, it is not necessary to run all three models if you use the pre-processed data to run our model. These three models are publicly available. If you want to run the pre-processing part as well, you need to download all the models and setup according to their instructions. We provided the download link and also our code where we used their model to generate a set of features.

A.2 Description & Requirements

To run our model (Table 3 in the paper), we only need some basic Python library and that is provided in the Python script. This part runs our model on pre-processed data. No additional requirements are required if you run our model on the pre-processed data. Data is also provided and all the instructions are available in the Readme file.

However, if we want to run the data pre-processing, you need to download Code2vec and CodeBERT model. Code2vec is implemented using the approach by `https://github.com/dcoimbra/dx2021`. In order to download, one needs to follow the instructions provided in their repository. Once downloaded, our part of code is available and the instructions are provided in the Readme. For CodeBERT, we utilized `https://github.com/microsoft/CodeXGLUE/tree/main/Code-Code/Defect-detection` following their instruction. We have provided our part of the code and instructions are provided in the Readme.

A.2.1 Security, privacy, and ethical concerns

None

A.2.2 How to access

The artifact can be accessed via following link: `https://github.com/SamihaShimmi/VulSim/tree/f08ab42dc8131b97887ee6e6d547df2a7915ee02`

A.2.3 Hardware dependencies

No additional hardware resource is required to run our model. We ran our model in Google Colab, using CPU. To pre-process data, when we utilized other models such as Code2vec, we used High-RAM mode.

A.2.4 Software dependencies

None

A.2.5 Benchmarks

We used Devign dataset from CodexGlue Benchmark that is available in `https://sites.google.com/view/devign`. We had to discard 71 records from the original benchmark and our dataset without those 71 records are available here `https://drive.google.com/drive/folders/1AGFr3Z3yfhwY5HYW3vvMSY5KGrVzFG7K?usp=sharing`. In order to run the model with pre-processed data, you can use the data available in our github repository. This dataset is only required if you want to run the pre-processing steps.

A.3 Set-up

All setup steps are embedded in corresponding scripts.

A.3.1 Installation

You have to download the repository from the provided GitHub link. No additional installation is required.

A.3.2 Basic Test

Since there is not any prerequisite to run our model, there is no basic test. You can run our decision tree-based classifier to reproduce Table 3.

A.4 Evaluation workflow

This section describes the basic workflow.

A.4.1 Major Claims

(C1): Our major claim is that our model outperforms existing State-Of-The-Art (SOTA) methods. Table 3 is the outcome of this claim.

A.4.2 Experiments

(E1): [Basic claim in pre-processed dataset (replicate table 3)] [5 mins max]: If you only want to generate the result on the hybrid model, it will only take approximately 2-5 minutes with the pre-processed data
How to: Just following the Readme file and using the appropriate dataset will be sufficient. All required data and the classifier code is available.
Preparation: No preparation is required. You can just run the decision tree based classifier on pre-processed data which are available in the GitHub repository.
Execution: You have to follow the python script sequentially provided in the Github repository.
Results: The last part of the python script gives the accuracy which is reported in Table 3.

(E2): [Basic claim without pre-processed data (replicate table 3 with pre-processing the data)] [2 human-hour + 120 hours approximately (if run separately for each model. If you run parallel, it will be 48-72 hours in CPU mode. If you have more advanced hardware, it will be less)]:
How to: We leveraged three existing models to preprocess our data. As we mentioned in the Section A.2.5, we used Devign dataset and discarded 71 records (download link is provided in both Section A.2.5 and Readme file). You have to download this dataset in order to preprocess.
Preparation: Code2vec and CodeBERT model should be downloaded using the following links: https://github.com/dcoimbra/dx2021 and https://github.com/microsoft/CodeXGLUE/tree/main/Code-Code/Defect-detection
Execution: Once you download the models, you can leverage the models using our provided scripts. We have three models namely Code2vec, CodeBERT, and SBERT. For each three models, in the Github link, we have provided our scripts. Name of the each file is given in the Readme files. You have to run two scripts for each model. The first one generates the embeddings and calculates cosine similarity. The second script applies our ranking formula and generates the features to feed our model. SBERT part takes less time. For code2vec and CodeBERT, generating the embedding and cosine similarity takes some time (48-72 hours for each model) to execute depending on the hardware. Once we generate the embeddings, we manually need to run some scripts step by step (following the Readme) to generate the features. Once we have all the features from three models, we merge 2 sets of features from each model and thus get the pre-processed data to feed to the decision tree based classifier. We need to follow the steps provided in E1 to regenerate Table 3.
Results: The last part of the python script gives the accuracy which is reported in Table 3.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.