
# USENIX Security '24 Artifact Appendix: Code is not Natural Language: Unlock the Power of Semantics-Oriented Graph Representation for Binary Code Similarity Detection

Haojie He[1], Xingwei Lin[2], Ziang Weng[1],
Ruijie Zhao[1], Shuitao Gan[3], Libo Chen[1], Yuede Ji[4], Jiashui Wang[2], and Zhi Xue[1]
[1]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University [2]Ant Group
[3]Laboratory for Advanced Computing and Intelligence Engineering [4]University of North Texas
*{sgfvamll, ziangweng, ruijiezhao, bob777, zxue}@sjtu.edu.cn*,
*{linyi.lxw, quhe}@antgroup.com*, *ganshuitao@gmail.com*, *yuede.ji@unt.edu*

## A Artifact Appendix

### A.1 Abstract

Our artifact contains the source code, data, results, and links to external resources (e.g., code and datasets published by previous work) for the experiments we conduct in this paper. We aim to show that HermesSim can achieve the claimed performance on the well-established dataset published by previous work.

### A.2 Description & Requirements

*[Mandatory] This section should list all the information necessary to recreate the same experimental setup you have used to run your artifact. Where it applies, the minimal hardware and software requirements to run your artifact. It is also very good practice to list and describe in this section benchmarks where those are part of, or simply have been used to produce results with, your artifact.*

#### A.2.1 Security, privacy, and ethical concerns

None.

#### A.2.2 How to access

The source code, analysis scripts, and their usage instructions are available at: https://github.com/NSSL-SJTU/HermesSim/tree/a8fee0d218519826dd4a7a9799077eff. We provide our datasets and all evaluation results at https://zenodo.org/records/10369788.

#### A.2.3 Hardware dependencies

We recommend using a PC with 16 GiB of main memory and a GPU with 10GiB of memory (NVIDIA GeForce RTX 3080 is our choice). Approximately 300 GiB of disk space is needed to store the dataset and (intermediate) results for this evaluation.

#### A.2.4 Software dependencies

We evaluate our methods with pytorch 1.13.1 with cuda 11.6. Other dependencies can be found in https://github.com/NSSL-SJTU/HermesSim/tree/a8fee0d218519826dd4a7a9799077eff/requirements.txt.

#### A.2.5 Benchmarks

We use the dataset published on https://github.com/Cisco-Talos/binary_function_similarity. We have pre-processed the dataset and provide the inputs suitable for our model on https://zenodo.org/records/10369788/files/inputs.tar.xz?download=1. Still, the evaluator and other researchers can choose to reproduce all pre-processing steps following the instructions on https://github.com/NSSL-SJTU/HermesSim/tree/a8fee0d218519826dd4a7a9799077eff/README.md#how-to-reproduce-the-experiments. Additionally, transferring the processing steps to other datasets can be done without significant modifications. This has been tested on a self-built real-world RTOS dataset.

### A.3 Set-up

#### A.3.1 Installation

- Clone our github repository, enter the project root and setup the python environment using the command provided in https://github.com/NSSL-SJTU/HermesSim?tab=readme-ov-file#0-setup.

- Make a sub-directory named `dbs`. Download and extract the dataset meta files and the lifted binary functions on https://zenodo.org/records/10369788/files/dbs.tar.xz?download=1 to the `dbs` sub-directory.
- Make a sub-directory named `inputs`. Download and extract the pre-processed model input on https://zenodo.org/records/10369788/files/inputs.tar.xz?download=1 to the `inputs` sub-directory.
- Make a sub-directory named `outputs`. Optionally, download and extract our experiment results on https://zenodo.org/records/10369788/files/outputs.tar.xz?download=1 to the `outputs` sub-directory.

#### A.3.2 Basic Test

At this point, the evaluator should be able to train the model. Test it with:

```
python model/main.py \
--config model/configures/e00_major_noinfer.json \
    --inputdir dbs \
    --dataset=one \
    --num_epochs=1
```

### A.4 Evaluation workflow

#### A.4.1 Major Claims

**(C1):** HermesSim can achieve the state-of-the-art on the well-established dataset published by previous work. This is proven by the experiment (E1) described in Section 5.2 whose results are reported in Table 1.

**(C2):** The semantics-oriented graph (SOG) representation proposed in this paper is more appropriate for binary code similarity detection than previous representations. This is proven by the experiment (E2) described in Section 5.3 whose results are reported in Table 2.

**(C3):** The multi-head softmax aggregator proposed in this paper is more efficient than previously used aggregator models. This is proven by the experiment (E3) described in Section 5.3 whose results are also reported in Table 2.

#### A.4.2 Experiments

**(E1):** *[HermesSim Evaluation] [15 human-minutes + 1 compute-hour + 1GB of disk space for results]: Retraining and testing HermesSim to generate data supporting C1.*

**Execution:** First, retrain HermesSim and then infer the testing dataset using:

```
python model/main.py --inputdir dbs \
    --config ./model/configures/e00_major.json \
    --dataset=one
```

This will cost about 1 compute-hour (with NVIDIA GeForce RTX 3080). And an output directory is automatically created under the `$HermesSimRoot/outputs/`. Second, evaluate and collect results with:

```
python \
postprocess/2.summarize_results/collect_stats.py \
    dbs/Dataset-1/pairs/experiments/ \
    outputs/Output-xxxx_yyyyyy
```

The directory name `Output-xxxx_yyyyyy` should be replaced with the output directory generated in the first step.

**Results:** A python notebook named `print_plot_results-custom.ipynb` is provided to ease the collection and review of experiment results (under the directory `postprocess/3.pp_results/`). Find the block titled with `E1`, replace the `EXPERIMENT_ROOT` constant with the output directory generated in the first step, and re-run that block. Due to randomness, the replicated results may not be exactly the same as those presented in the paper (results presented in the paper are the average of 10 independent runs). Typically, the error is within the range of 1%.

**(E2):** *[Ablation Study - Representation Part] [1 human-hour + 6.5 compute-hour + 5GB of disk space for results]* All steps are the same as E1 except that in the first step, the evaluator should invoke the `model/main.py` with a different configure file `./model/configures/e02_repr.json`. The experiment for CFG-PalmTree representation is not reproduced in this step since it only supports the x64 architecture and needs special setups.

**(E3):** *[Ablation Study - Aggregator Part] [1 human-hour + 3 compute-hour + 1.5GB of disk space for results]* All steps are the same as E1 except that in the first step, the evaluator should invoke the `model/main.py` with a different configure file `./model/configures/e02_aggrs.json`.

### 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/.