Practical Binary Code Similarity

Detection with BERT-based

Transferable Similarity Learning

|  |  |  |  |
| --- | --- | --- | --- |
| Sunwoo Ahn | Seonggwan Ahn | Hyungjoon Koo | Yunheung Paek |
| Department of Electrical | Department of Electrical | Department of Computer | Department of Electrical and |
| and Computer Engineering | and Computer Engineering | Science and Engineering | Computer Engineering |
| Seoul National University | Seoul National University | Sungkyunkwan University | Seoul National University |
| Seoul, Korea | Seoul, Korea | Suwon, Korea | Seoul, Korea |

Binary Code Similarity Detection (BCSD)

* BCSD Problem



|  |  |  |  |
| --- | --- | --- | --- |
| machine code | Proximity? | machine code |  |
|  |  |
| snippet A |  | snippet B |  |

* Many applications
  + Code clone detection
  + Malware detection
  + Malware family classification
  + Known vulnerability discovery
  + Code patching verification

Challenges

* Useful information is unavailable in a binary
  + e.g., variable name, structure, type, class hierarchy, etc.
* Binaries that have identical semantics can vary
  + compiler configuration, architecture, obfuscation, etc.
* Halting problem
  + Undecidable to prove

the functional equivalency of two arbitrary programs

Existing Works

* Recent advances employ

neural network with Siamese architecture

|  |  |
| --- | --- |
| **Model** | **Architecture** |
|  |  |
| Gemini | GNN, Siamese NN |
|  |  |
| InnerEye | word2vec, LSTM, Siamese NN |
|  |  |
| Asm2Vec | PV-DM |
|  |  |
| PalmTree | BERT, GNN, Siamese NN |
|  |  |
| DeepSemantic | BERT, Softmax classifier |
|  |  |

Existing Works

* Distance/loss function affects Siamese network (Marcelli et al., USENIX '22)

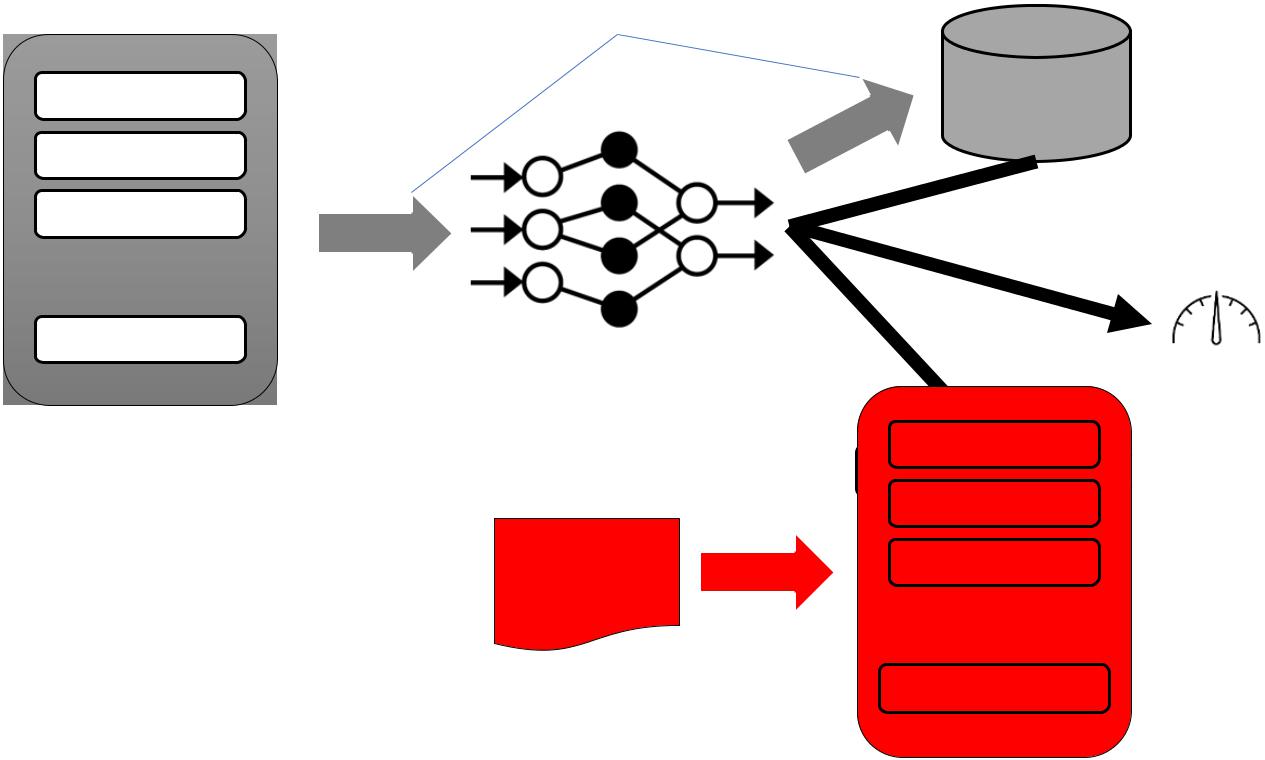
|  |  |  |
| --- | --- | --- |
| **Model** | **Distance function** | **Loss function** |
|  |  |  |
| Gemini | Cosine distance | Contrastive loss |
|  |  |  |
| InnerEye | Cosine distance | Contrastive loss |
|  |  |  |
| Asm2Vec | Cosine distance | Log probability |
|  |  |  |
| PalmTree | Cosine distance | Contrastive loss |
|  |  |  |
| DeepSemantic | None | Cross entropy |
|  |  |  |

* Scalar value  oversimplification

Problem

* We question existing work in a realistic scenario

①



function1

function2

function3

|  |  |
| --- | --- |
| … |  |
| function\_n | BCSD Model |

Function

embeddings of

one’s interest

Database

②

Query

Binary

function1

Query function

function2

function3

|  |
| --- |
| … |

Similarity

score

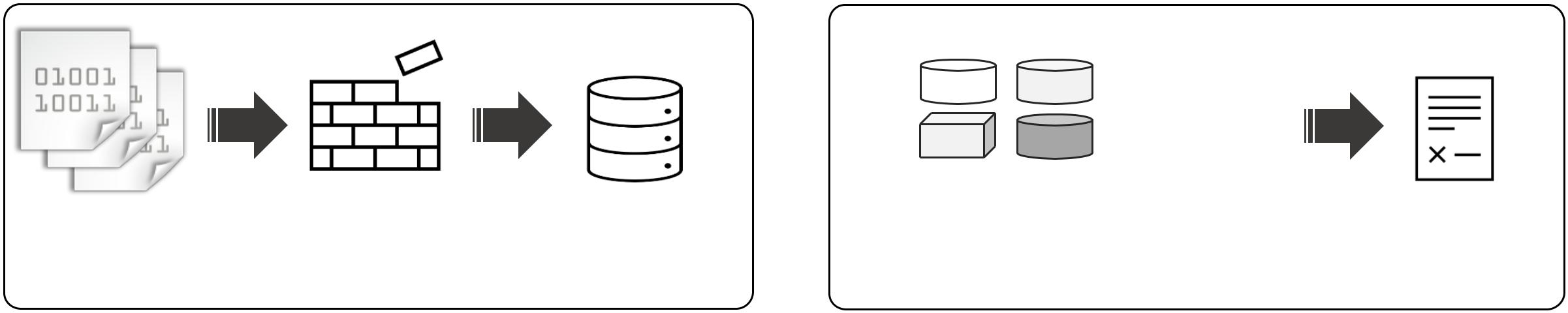
function\_m

Our Main Approach

* Goal: improve performance for unseen dataset
* Transferable similarity learning (BERT-based)
  + Learning a relationship btw instructions with pre-training
  + Repeatedly showing good performance on an assembly language
* Better similarity detection: learning a weighted distance vector with a binary cross entropy
* Weighted distance  relationships are represented in a vector

BinShot

Pre-processer Fine-tuner



**Similar**

**Dissimilar**

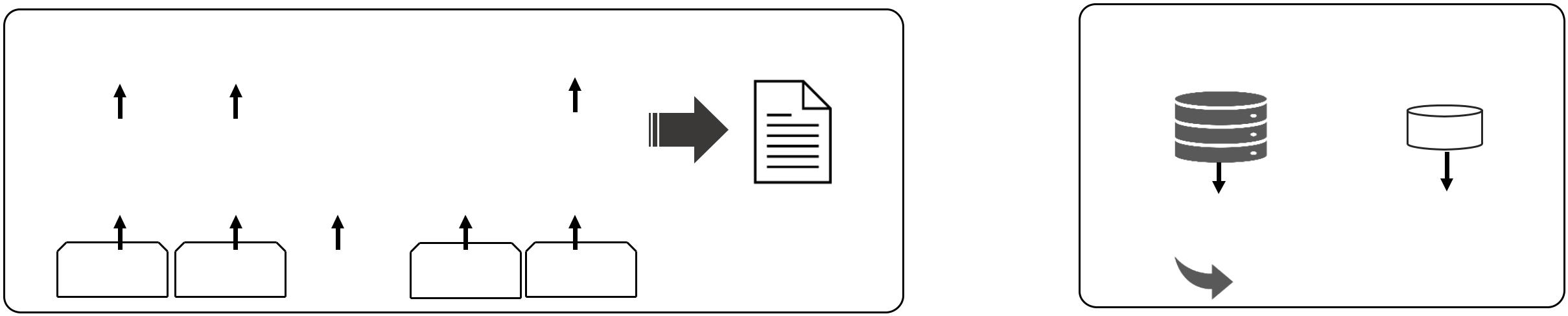
**Executables Disassembled Normalized**

**(Corpus)** **Functions** **Functions**

|  |  |  |
| --- | --- | --- |
| Downstream Layers | **Fine-tuned** |  |
|  | **BERT Model** |  |
|  |  |

* **Preprocessing for Training Preparation**③ **Building a Special Model for Code Similarity**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pre-trainer | | | | | | | | | | | | | | | | | | | | | | | | | | |  | Predictor | |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  | **Logits** | | | | | | | | | | | | | | |  | Function | | Target |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **…** | |  | Embeddings | | Function |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | Feed-forward Neural Network | | | | | | | | | | | | | | | | | | | | | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  | BERT | | | | | | | | | | | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | **Pre-trained** |  | Downstream Model | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | [SOS]Ins 0 [MASK] Ins 2[EOS] | | | | | | | | | | | | | | | | | | | | | | | | |  |
|  |  | **BERT Model** |  | Prediction: similar? | |  |
| **② Building a Generic Model for Assembly** | | | | | | | | | | | | | | | | | | | | | | | | | | |  | ④ **Detecting Similarity** | | |  |

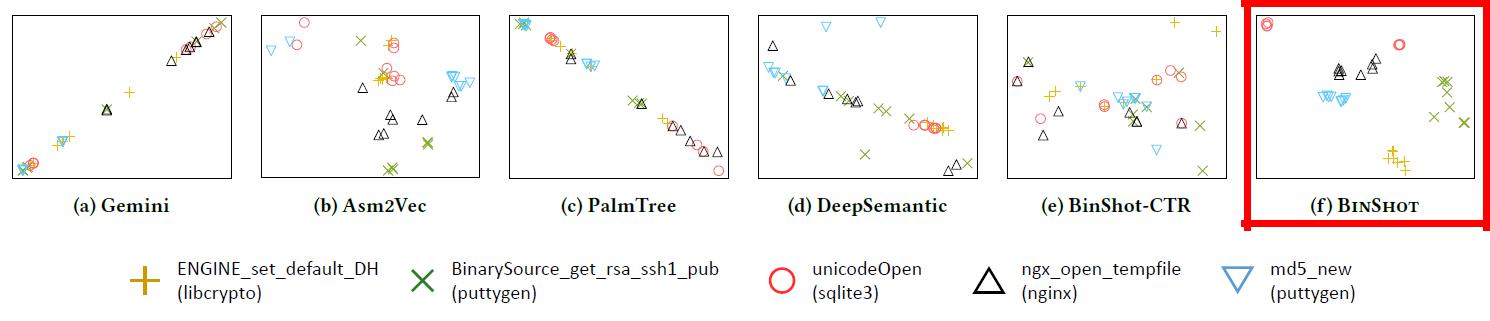
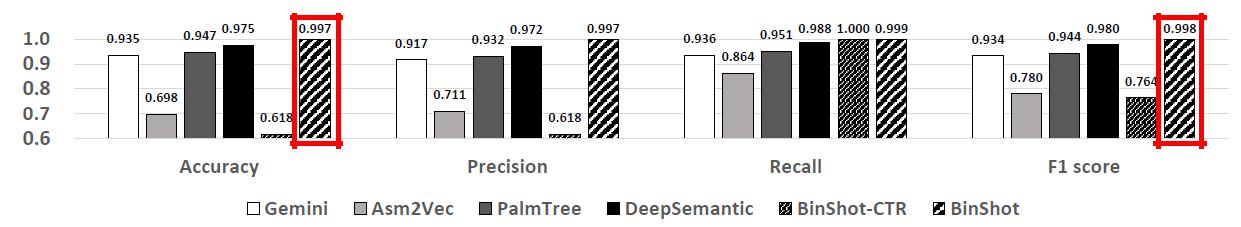


Experimental Setup

* Dataset
  + Compiled with 2 compiler (gcc, clang) & 4 optimization (O0-O3)
  + 1,400 binaries in total
    - GNU utilities – binutils, coreutils, diffutils, findutils
    - SPEC2006, SPEC2017
    - 11 Real-world programs (BusyBox, Libgmp, …)
* Baseline models:
* Gemini, Asm2vec, PalmTree, DeepSemantic
* BinShot-CTR, BinShot

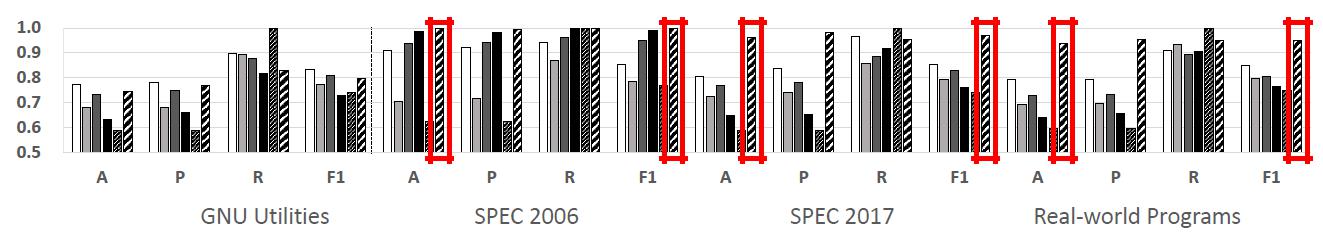
Evaluation - Effectiveness

* Evaluate whole dataset
* t-SNE visualization



Evaluation - Transferability

* Trained with SPEC 2006

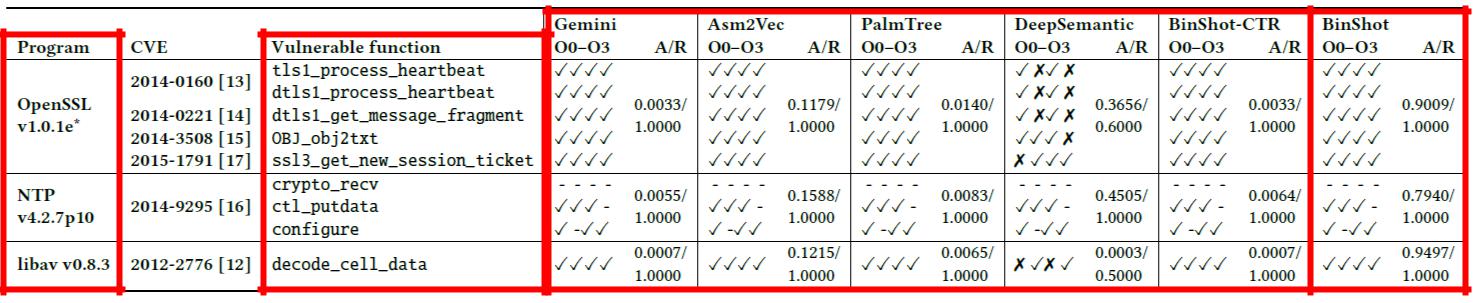


Evaluation – Vulnerable Function Detection

* Realistic scenario setup
  + Database contains

vulnerable function embeddings

* + Query binary is stripped
  + Goal: find a vulnerable function from a query binary



Evaluation – Runtime Efficiency

* Runtime efficiency
  + Exp1 - Each function pair
  + Exp2 - 82300 function pairs (100 in database, 823 in query binary) with our predictor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Gemini | Asm2Vec | PalmTree | DeepSemantic | BinShot-CTR | BinShot |
|  |  |  |  |  |  |  |
| Exp1 (ms) | 0.10 | 81.94 | 1.33 | 1.34 | 1.30 | 1.32 |
|  |  |  |  |  |  |  |
| Exp2 (s) | 1.16 | 6,734.66 | 29.03 | 1.51 | 1.45 | 1.54 |
|  |  |  |  |  |  |  |

Discussions & Limitations



* Mangled Names
* Function inlining
* Code obfuscation and other code constructs
* Rarely appeared instructions

Wrap-up

* Learning a weighted distance with a binary cross entropy improves robustness against unseen function pairs
* Superiority of BinShot
  + effectiveness, practicality (transferability & runtime)
* The other models but ours shows poor performance in a realistic scenario
* Open source project: https://github.com/asw0316/binshot

Thanks!