Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection

Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, Dawn Song. CCS 2017

Keywords: **Binary code similarity detection**

LyuJiuyang, Dec 16th, 2021.

Motivation

Existing approaches rely on approximate graph matching algorithms, which are inevitably slow and sometimes inaccurate, and hard to adapt to a new task.

Contributions

- propose the first neural network-based approach to generating embeddings for binary functions;
- propose a novel approach to train the embedding network using a *Siamese network* so that a pre-trained model can generate embedding to be used for similarity detection;
- propose a retraining approach so that the pre-trained model can take additional supervision to adapt to specific tasks;
- implement a prototype called Gemini, which can achieve a higher AUC.

Task Definition

Given a binary function of interest, we would like to examine a large corpus of binary functions and quickly and accurately identify a list of candidates that are semantically equivalent or similar to the function of interest.

Current Work

Pairwise Graph Matching

The approach *[IEEE-SSP 2015 Cross-Architecture Bug Search in Binary Executables.]* extracts input-output pairs for each basic block as its feature (or label) in the control flow graph, and then it performs graph matching.

- time-consuming, small codebooks, low accuracy

Methodology

Code Similarity Embedding Problem

Giving two binary functions f_1, f_2, calculate the similarities of them:

$$Sim(\phi(f_1), \phi(f_1))$$

Using Neural Network to estimate ϕ.

Solution Overview

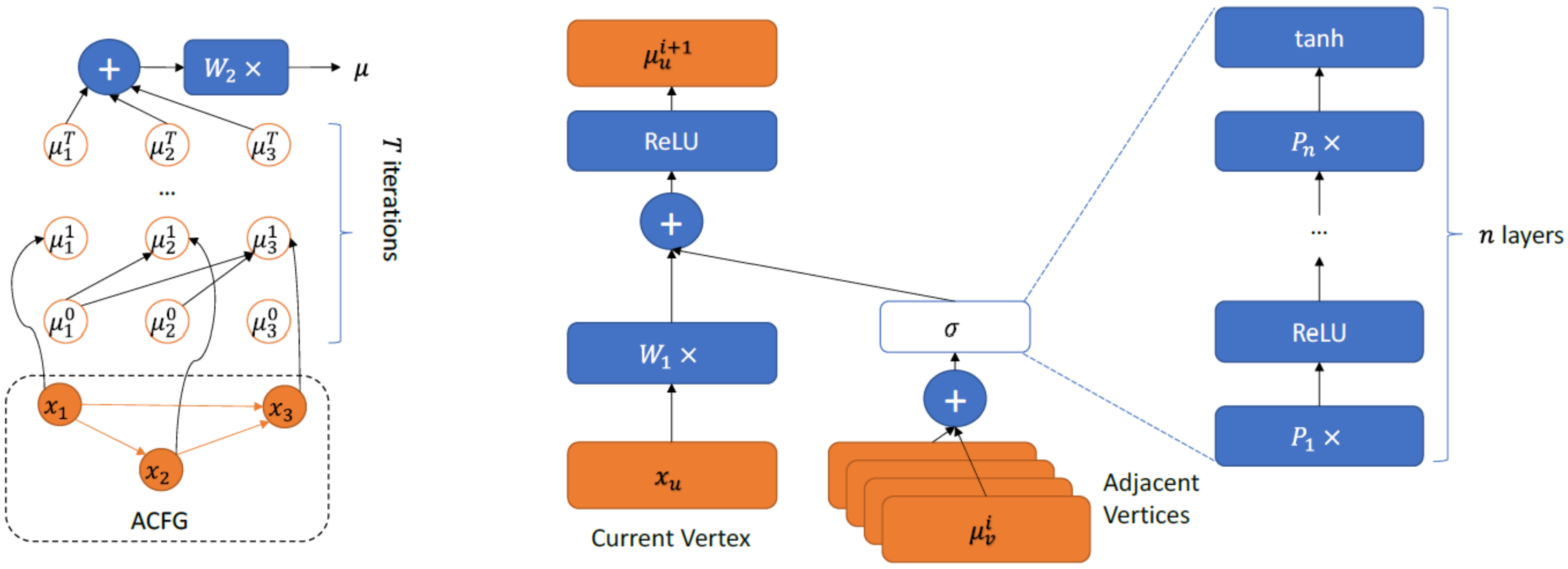
Train ϕ to do well on differentiating the similarity between two input ACFGs. They design a Siamese architecture and embed the graph embedding network Structure2vec into it.

Graph Embedding Network

Denote an ACFG as $g = \langle V, \varepsilon \rangle$ (Vertice and edges).

Structure2vec

> similar with the aggregation in NGCF.



(a) Graph Embedding Network Overview

(b) One Layer (Iteration) of the Graph Embedding Network

Figure 3: Graph Embedding Network

Structure2vec network will init each embedding as 0 and update the embeddings at each iteration as

$$\mu_v^{(t+1)} = F(x_v, \sum_{u \in N(v)} \mu_u^{(t)}), \forall v \in V$$

$$F(x_v, \sum_{u \in N(v)} \mu_u) = \tanh(W_1 x_v + \sigma(\sum_{u \in N(v)} \mu_u))$$

x_v means vertex-specific features, $N(v)$ denotes the set of neighbors of vertex

σ is a n-layer full-connected NN.

$$\sigma(l) = \underbrace{P_1 \times ReLU(P_2 \times \ldots ReLU(P_n l))}_{n\ levels}$$

Algorithm 1 **Graph embedding generation**

1: **Input:** ACFG $g = \langle \mathcal{V}, \mathcal{E}, \overline{x} \rangle$
2: Initialize $\mu_v^{(0)} = \overline{0}$, for all $v \in \mathcal{V}$
3: **for** $t = 1$ **to** T **do**
4:     **for** $v \in \mathcal{V}$ **do**
5:         $l_v = \sum_{u \in \mathcal{N}(v)} \mu_u^{(t-1)}$
6:         $\mu_v^{(t)} = \tanh(W_1 x_v + \sigma(l_v))$
7:     **end for**
8: **end for**{fixed point equation update}
9: return $\phi(g) := W_2(\sum_{v \in \mathcal{V}} \mu_v^{(T)})$

Siamese Architecture (Cosine similarity)

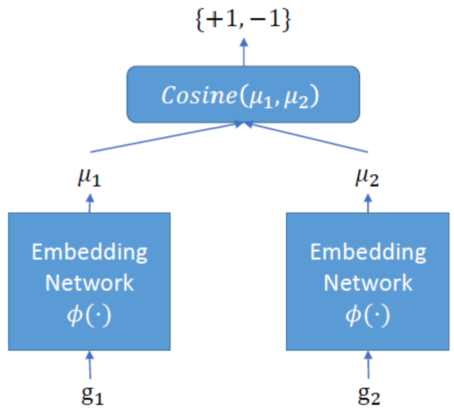


Figure 4: Siamese Architecture

$$Sim(g_i, g'_i) = \cos(\phi(g), \phi(g')) = \frac{\langle g_i, g'_i \rangle}{\|\phi(g)\| \cdot \|\phi(g')\|}$$

+1 denotes ACFG g_1 is similar to g_2. Two embedding networks share the same set of parameters.

Objective function

$$\min_{W_1, P_1, \ldots, P_n, W_2} \sum_{i=1}^{K} (Sim(g_i, g'_i) - y_i)^2$$

using SGD as optimizer.

Other

- Appreciate the improvement on Graph Embedding Network. It is worth learning.