A Cross-Architecture Instruction Embedding Model for Natural Language Processing-Inspired Binary Code Analysis

Kimberly Redmond, Lannan (Lisa) Luo, Qiang Zeng
University of South Carolina
Closed-Source Software

- When using proprietary software, often we are only left with binaries
- Software on embedded devices (*firmware*) is usually closed-source

- *Binary code analysis* is an important method for analyzing programs through their binaries. It can be applied to tasks, such as code plagiarism detection, vulnerability discovery, and malware detection
Software is increasingly cross-compiled for various architectures
Our Insight

Binary code analysis can be approached by borrowing ideas and techniques of *Natural Language Processing*.

**NLP:**
- words $\rightarrow$ word embeddings (i.e., high-dimensional vectors)

**NLP-inspired binary code analysis:**
- instructions are regarded as words
- instruction $\rightarrow$ instruction embeddings
Background: Word Embeddings

- Word embeddings are **high-dimensional vectors** that **encode word meanings**

- One-hot encoding: Given a dictionary of 100 words, each word occupies one dimension out of 100 in an all-0 vector

\[
\begin{align*}
\text{Cat} &= [1 \ 0 \ 0 \ 0 \ 0 \ …] \\
\text{Bird} &= [0 \ 0 \ 1 \ 0 \ 0 \ …] \\
\text{Dog} &= [0 \ 1 \ 0 \ 0 \ 0 \ …] \\
\text{Pig} &= [0 \ 0 \ 0 \ 1 \ 0 \ …]
\end{align*}
\]

But this does not tell us how words are similar or different
Background: Word Embeddings

• To reflect what words *mean*, dimensions will instead encode patterns of how words are distributed across texts

• *Insight:* if two words tend to appear in the *same* contexts, then the two words probably share the same meaning
Background: **Multilingual** Word Embeddings

- Multiple human languages
- Various multilingual NLP tasks
- *Multilingual* word embedding models learns word embeddings such that: *similar words in different human languages have similar embeddings*
Cross-Architecture Binary Code Analysis

NLP-inspired binary code analysis:
- instructions are regarded as words
- instruction → instruction embeddings

Cross-architecture binary code analysis:
- instruction → cross-architecture instruction embeddings
- similar instructions from different arch. have similar embeddings
Motivation

All ARM and x86 instructions; if the embeddings are trained separately

All ARM and x86 instructions; if the embeddings are trained jointly
Potential Applications

- Code similarity comparison:
  - Summing up all the embeddings of instructions in a function/basic block, and using the sum to represent the function/basic block for similarity comparison
  - Some previous work based on deep learning (e.g., *InnerEye*[NDSS’19], *Arm2Vec*[S&P’19], *i2V-RNN*[BAR’19]) use complex neural network models, such as LSTM, structure2vec

- Transferability:
  - Training a classifier using the code of x86, and directly applying the classifier to the code of ARM
  - ……
Our Training Approach

- We adopt the BiVec model, a multilingual word embedding model.
- Finding the alignment links: simply assume \textit{linear alignments}
  - Each instruction in one sequence $M$ at position $i$ is aligned to the instruction in another sequence $N$ at position $\lceil i \times \frac{|N|}{|M|} \rceil$
  - E.g., $M = \{u1, u2, u3, u4\}$, $N = \{v1, v2, v3\}$, the alignment links: $u1\leftrightarrow v1$; $u2\leftrightarrow v2$; $u3\leftrightarrow v3$; $u4\leftrightarrow v3$;
Evaluation

- **Dataset:** 202,252 semantically similar basic blocks generated by our another work [1]

- Two types of experiments:
  - Instruction similarity tasks:
    - Mono-architecture instruction similarity task
    - Cross-architecture instruction similarity task
  - Cross-architecture basic-block similarity comparison task

Mono-Architecture Instruction Similarity Task

- 100 instruction pairs were randomly chosen and labeled (50 similar, 50 dissimilar). This was determined by *opcodes*.
- Cosine similarity
  - ARM AUC = 0.82
  - X86 AUC = 0.74
Cross-Architecture Instruction Similarity Task

- 50 pairs of instructions across architectures were randomly chosen and labeled (25 similar, 25 dissimilar). Again, *opcodes* were used to decide this.
  - AUC = 0.72

- The results are good, but an advanced way of finding alignment links between instructions would improve the results.

![Cross-Architecture Instruction-Level Similarity](image-url)
Cross-Architecture Basic-Block Similarity Test

- 90% of similar basic block pairs for training
- 10% of similar block pairs and another 20,633 dissimilar pairs (selected from [1]) for testing
- Summation of all instruction embeddings to represent a block
  - AUC = 0.90
- Recent work (such as Gemini in CCS’17) uses manually selected features to represent a basic block; a SVM classifier based on such features can only achieve AUC = 0.85

T-SNE Visualizations

Visualization of five ARM and x86 instruction pairs. A blue circle and red triangle represent an ARM and x86 instruction, respectively.
Summary

• The first work discusses cross-architecture instruction embedding models.

• We build the cross-architecture instruction embedding model, such that similar instructions, regardless of their architectures, have embeddings close together in the vector space.

• We conduct various experiments to evaluate the quality of the learned instruction embeddings.

• The proposed model may be applied to many cross-architecture binary code analysis tasks, such as vulnerability finding, malware detection, and plagiarism detection.
https://github.com/nlp-code-analysis/cross-arch-instr-model

Thank you