Transforming Generic Coder LLMs to Effective Binary Embedding Models for Similarity Detection

Litao Li*, Leo Song*, Steven H. H. Ding, Benjamin C.M. Fung, Philippe Charland

Queen's University, McGill University, Defence R&D Canada

*Equal Contributions

Introduction

Binary code similarity detection (BCSD) is a challenging and practical problem in the software security landscape. Existing deep-learning methods train on limited compiler settings and thus lack generalizability. We propose EBM, a multi-component framework to transform coder LLMs to binary code embedding models. The components include data augmentation, translation-style causal learning, LLM2Vec, and cumulative GTE loss. Our major contributions are:

- We tackle different compiler settings of BCSD, including cross-optimization, cross-architecture, and cross-obfuscation, with a general and effective similarity retriever.
- To address the limitations of LLMs for assembly code, we finetune a generic LLM to a binary code-specific model by enabling the power of LLMs with domain-specific training for adaptation.
- Our experiments indicate that EBM outperforms all baselines including embedding models and existing BCSD methods. The ablation study further shows that all components significantly improve the results.

Methodologies

1. Enabling Translation and Structure Awareness

To clean the data, all addresses, strings, and bytes are replaced by special tokens: **addr**, **byte**, **str**. Secondly, we flatten the original data structure of assembly code, which contains basic blocks and instruction streams, into a single sentence. We propose to simply add a special **BLK** token in between basic blocks to increase structure awareness during training.

2. Binary Translation Continual Training

For a pair of binaries (X1, X2), we concatenate them into a single sequence by X = [X1, (binary settings of X1), (binary settings of X2), X2]. The model is trained with next token prediction. The goal is to enable cross-architecture translation

3. LLM2Vec

We enable the bidirectional attention of an autoregressive model. The model is then fine-tuned by masked next token prediction (MNTP).

$$\mathcal{L}_{\text{MNTP}} = -\sum_{i}^{masked} P(x_i) \log P(\hat{x}_i | x_1, ..., x_n)$$

Equation 1: MNTP loss

4. cGTE

The GTE (general text embedding) loss (Equation 2) is calculated from the cumulative embeddings from distributed devices and async batches such that every gradient update is based on every available contrastive pair.

$$q = q^{(1)}||q^{(2)}||...||q^{(n)}$$
$$k = k^{(1)}||k^{(2)}||...||k^{(n)}$$
$$\mathcal{L}_{\text{cGTE}} = \text{GTE}(q, k)$$

Equation 2: cGTE loss

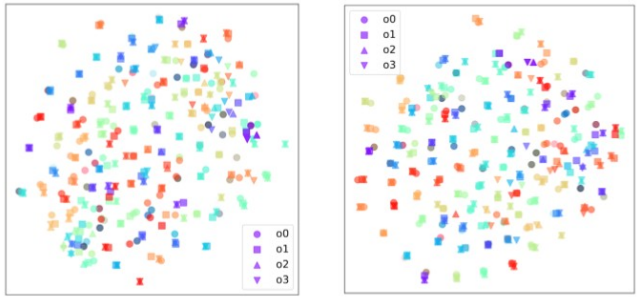
Experiments & Results

Two datasets are used for training and evaluation. The first one is a collection of libraries [Ding et al., 2019]. We manually compile the files using different optimization levels, compilers, architectures, and obfuscations. The second dataset is BinaryCorp [Wang et al., 2022] where only different optimizations are evaluated. The evaluation is conducted as a retrieval task, where given a query function, EBM outputs the most similar function from a pool of functions. The metrics used are Mean Reciprocal Rank (MRR) and Recall@1. We conduct more than 31 groups of embedding retrieval experiments for 10 models.

Models	MRR					
	O0,O3	O0,O1	O0,O2	O1,O3	O2,O3	Avg.
SAFE	0.189	0.189	0.200	0.218	0.171	0.193
PalmTree	0.023	0.020	0.019	0.314	0.878	0.251
Asm2Vec	0.444	0.494	0.460	0.535	0.563	0.499
OrderMatters	0.006	0.006	0.008	0.006	0.006	0.006
GraphCodeBERT (125M)	0.636	0.757	0.673	0.792	0.920	0.756
CodeT5+ (110M)	0.604	0.650	0.629	0.830	0.893	0.721
Qwen2.5-Emb (1.5B)	0.569	0.648	0.573	0.773	0.907	0.694
Qwen2.5-Coder (1.5B)	0.758	0.881	0.807	0.864	0.936	0.849
CodeGemma (2B)	0.763	0.888	0.833	0.866	0.931	0.856
EBM (0.5B)	**0.850**	**0.942**	**0.902**	**0.933**	**0.955**	**0.916**

Models	MRR			
	Arm, x64	PowerPC, x64	MIPS, x64	Avg.
SAFE	0.239	0.187	0.196	0.208
PalmTree	0.037	0.036	0.018	0.031
Asm2Vec	0.242	0.293	0.417	0.317
OrderMatters	0.007	0.007	0.007	0.007
GraphCodeBERT (125M)	0.067	0.269	0.495	0.277
CodeT5+ (110M)	0.056	0.303	0.462	0.274
Qwen2.5-Emb (1.5B)	0.039	0.059	0.409	0.169
Qwen2.5-Coder (1.5B)	0.256	0.481	0.548	0.428
CodeGemma (2B)	0.293	0.581	0.548	0.474
EBM (0.5B)	**0.783**	**0.792**	**0.859**	**0.811**

Table 1: a snapshot of MRR for cross-optimization and cross-architecture evaluations



(c) GraphCodeBERT (e) EBM

Figure 1: a snapshot of t-SNE visualization for embedding distributions.

Ablations

We abalte each component of our proposed methodologies to prove their necessity. Figure 2 shows the difference in MRR for the ablation. Every component contributes to learning different compiler settings and overall improves the efficacy. Among them, LLM2Vec is the key factor that can transfer the knowledge of LLM and further improves embedding semantics.

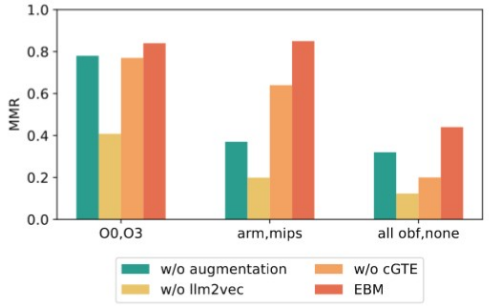


Figure 2: The performance comparison by ablating different components of our Methodologies.

Discussions and Limitations

EBM is a flexible framework that finetunes any LLMs for effective BCSD. It provides a generalizable and robust training paradigm that works for diverse compiler configurations. Our experiment and ablation study illustrates that the design of EBM achieves significant uplift from existing methods. Some limitations of our work include the lack of evaluation on downstream tasks and other bigger LLMs.

References

Steven H.H. Ding et al. (2019). "Asm2vec: Boosting static representation robustness for binary clone search against code obfuscation and compiler optimization" In: 2019 IEEE Symposium on Security and Privacy (sp), IEEE, pp. 472–489

Hao Wang et al. (2022). "Jtrans: Jump-aware transformer for binary code similarity detection" In: Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis, ACM, pp. 1-13