Poster: Comprehensive Comparisons of Embedding Approaches for Cryptographic API Completion

Ya Xiao¹, Salman Ahmed¹, Xinyang Ge², Bimal Viswanath¹, Na Meng¹, Danfeng (Daphne) Yao¹
¹Computer Science, Virginia Tech, Blacksburg, VA
²Microsoft Research, Redmond, WA
{yax99,ahmedms,vbimal,nm8247,danfeng}@vt.edu, aegiryy@gmail.com

ABSTRACT
In this paper, we conduct a measurement study to comprehensively compare the accuracy of Cryptographic API completion tasks trained with multiple API embedding options. Embedding is the process of automatically learning to represent program elements as low-dimensional vectors. Our measurement aims to uncover the impacts of applying program analysis, token-level embedding, and sequence-level embedding on the Cryptographic API completion accuracies. Our findings show that program analysis is necessary even under advanced embedding. The results show 36.10% accuracy improvement on average when program analysis preprocessing is applied to transfer byte code sequences into API dependency paths. The best accuracy (93.52%) is achieved on API dependence paths with embedding techniques. On the contrary, the pure data-driven approach without program analysis only achieves a low accuracy (around 57.60%), even after the powerful sequence-level embedding is applied. Although sequence-level embedding shows slight accuracy advantages (0.55% on average) over token-level embedding in our basic data split setting, it is not recommended considering its expensive training cost. A more obvious accuracy improvement (5.10%) from sequence-level embedding is observed under the cross-project learning scenario when task data is insufficient. Hence, we recommend applying sequence-level embedding for cross-project learning with limited task-specific data.

ACM Reference Format:
Ya Xiao¹, Salman Ahmed¹, Xinyang Ge², Bimal Viswanath¹, Na Meng¹, Danfeng (Daphne) Yao¹. 2022. Poster: Comprehensive Comparisons of Embedding Approaches for Cryptographic API Completion. In 44th International Conference on Software Engineering Companion (ICSE '22 Companion), May 21–29, 2022, Pittsburgh, PA, USA. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3510454.3528645

1 INTRODUCTION
API completion aims to predict the next API method given the previous code context, which is an important building block for many software engineering tasks. When training a neural network for API completion, code embedding is a key step. Code embedding refers to the process of transforming program elements to continuous vectors [1, 4]. This transformation is important, as subsequent model training and inference are performed on the embeddings.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
ICSE ’22 Companion, May 21–29, 2022, Pittsburgh, PA, USA © 2022 Copyright held by the owner/author(s).
https://doi.org/10.1145/3510454.3528645

We explore four research questions through our experiments. RQ1: What is the impact of program analysis preprocessing used with token-level embedding on cryptographic API completion accuracy? To answer this question, we compare the token-level embeddings, byte2vec, slice2vec, and dep2vec in API completion tasks. byte2vec is the embedding trained on byte code sequences. slice2vec is the embedding on program slices while dep2vec is the
embedding on API dependence paths. They differ in the program analysis preprocessing choices. We train identical LSTM models with different embedding options as inputs to achieve the cryptographic API completion. Table 1 shows the results. We observe the Finding 1 which answers our RQ1.

Table 2: Accuracy of the next API completion with or without sequence-level embedding (pretrain).

<table>
<thead>
<tr>
<th>Model Size</th>
<th>Byte Code</th>
<th>Slices</th>
<th>Dependence Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trans. + byte2vec (w/o. pretrain)</td>
<td>byteBERT (w. pretrain)</td>
<td>Trans. + dep2vec (w/o. pretrain)</td>
</tr>
<tr>
<td>Small</td>
<td>44.38%</td>
<td>45.21%</td>
<td>83.37%</td>
</tr>
<tr>
<td>Base</td>
<td>56.76%</td>
<td>57.59%</td>
<td>84.80%</td>
</tr>
</tbody>
</table>

Finding 1: For Crypto API completion with token-level embeddings, program analysis significantly improves the accuracy by 36.10% on average.

RQ2: What is the impact of applying token-level embedding on cryptographic API completion accuracy? To answer this question, we compared token-level embeddings with the baseline version, one-hot vectors, under identical program analysis preprocessing. Therefore, we have three groups of comparison between token-level embeddings and one-hot vectors, on byte code sequences, program slices, and API dependence paths, as shown in Table 1. We summarize our finding for RQ2 as follows.

Finding 2: For Crypto API completion on program slices and API dependence paths, token-level embedding achieves average accuracy improvement by 12.02% and 3.97%, respectively, compared with one-hot vectors.

RQ3: What is the impact of program analysis preprocessing used with sequence-level embedding on cryptographic API completion accuracy? We answer this question by comparing the sequence-level embeddings byteBERT, sliceBERT, and depBERT in cryptographic API completion. byteBERT, sliceBERT, and depBERT are the sequence-level embeddings trained on byte code sequences, program slices, and API dependence paths, respectively. We pretrain them on our Java cryptographic code by applying the masked language modeling on Transformer models. We observe the following finding from Table 2.

Finding 3: For Crypto API completion with sequence-level embedding, program analysis makes substantial accuracy improvement of 40.90% on average.

RQ4: What is the impact of applying sequence-level embedding on cryptographic API completion accuracy? To show the benefit of sequence-level embeddings, we compare the cryptographic API completion with or without a sequence-level embedding, byteBERT, sliceBERT, and depBERT are compared with the identical but unpretrained neural network (i.e., Transformer [8]). We perform the same task-specific training for them. Table 2 suggests the following finding for RQ4.

Finding 4: Despite the slight accuracy improvement (0.55% on average), sequence-level embedding is not the first recommended strategy to improve the Crypto API completion, compared with program analysis and a larger model.

Significance of research contributions. Our work provides the first quantitative and systematic comparison of the prediction accuracy of multiple API embedding approaches for neural network based code completion. Our rigorous experiments provide new empirical results that have not been previously reported, including how various domain-specific program analyses improve data-driven predictions. These quantitative findings, together with the new cryptographic API benchmark, help guide and design more powerful and accurate code completion solutions, leading to high quality and low vulnerability software projects in practice. In addition, our measurement methodology can be generalized to other types of APIs, beyond the specific cryptographic setting.

ACKNOWLEDGMENT

This work has been supported by the National Science Foundation under Grant No. CNS-1929701.

REFERENCES