Exploring API Embedding for API Usages and Applications

Trong Duc Nguyen, Anh Tuan Nguyen, Hung Dang Phan, Tien N. Nguyen

2017-39th International Conference on Software Engineering

Presented By: Saksham Gupta
saksham@vt.edu

Github Repository
Google Slides Link

Exploring API Embedding for API Usages and Applications

OVERVIEW

1. Aim of the paper
2. Background
3. Approach
   a. API2VEC
      i. Research Questions (Characteristics)
      ii. Research Findings
   b. API2API
      i. Research Findings
      ii. Ablation Study
4. Migration tool (Phrasal)
5. Related Work
6. Discussion and Questions
7. References

Exploring API Embedding for API Usages and Applications

A
I
M

Problem Statement

To study the characteristics of Word2Vec vectors called API2VEC or API embeddings for the API elements within the API sequence in source code.

B A C K G R O U N D

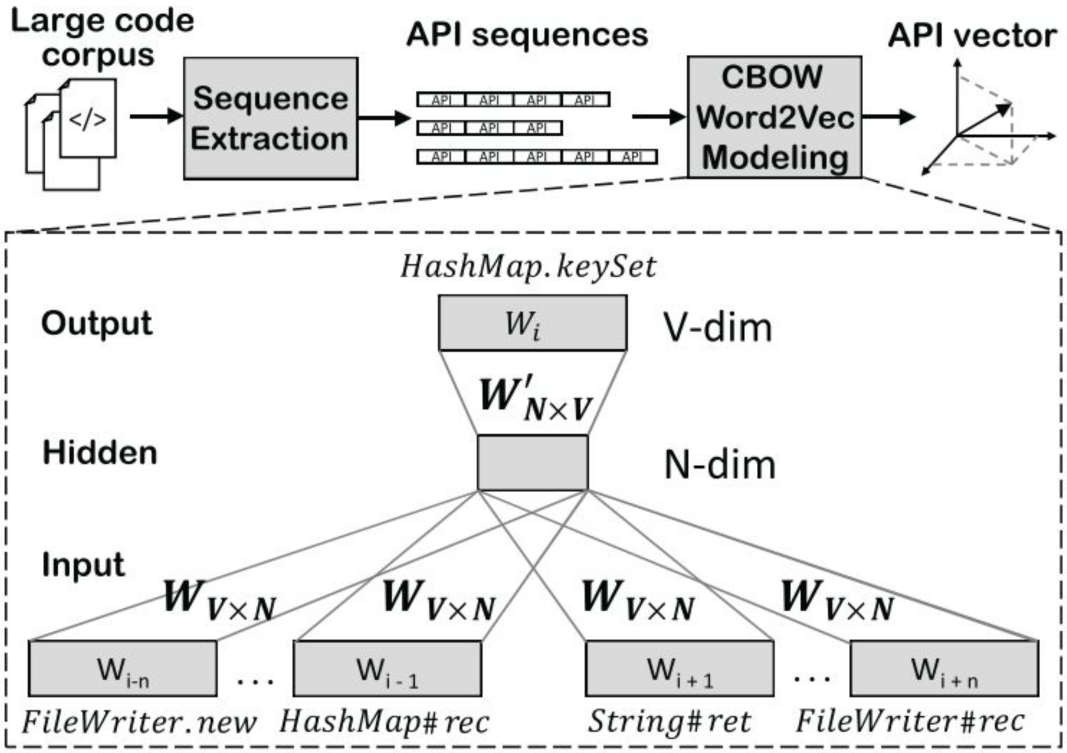
Exploring API Embedding for API Usages and Applications

Word2Vec

- A class of Neural Networks Model
- For each unique word produces a vector in a continuous space where linguistic context of words can be observed
- Encodes the contexts of surrounding words into vectors

$$V(w_i) = \frac{1}{2n}(w_{(i-n)} + ... w_{(i-1)} + w_{(i+1)} + ... + w_{(i+n)}).W_{V \times N}$$

- Training Criteria:
    - I/P to hidden weight matrix and hidden to O/P weight matrix results in $w = w_i$



Nguyen T. D., Nguyen A. T.,Phan H. D., and Nguyen T. 2017. Exploring API embedding for API usages and applications

Exploring API Embedding for API Usages and Applications

A
P
P
R
O
A
C
H

API2VEC: Research Questions

RQ1: *In a vector space for the APIs in usages, do nearby vectors represent the APIs that have similar usage contexts?*

Motivation: It has been shown that in the Word2Vec vector space for texts, the nearby vectors are the projected locations of the words [1] that have been used in the similar contexts consisting of similar surrounding words.

RQ2: *Can vector offsets in API2VEC capture similar usage relations (i.e.,co-occurring relations among APIs in usages)?*

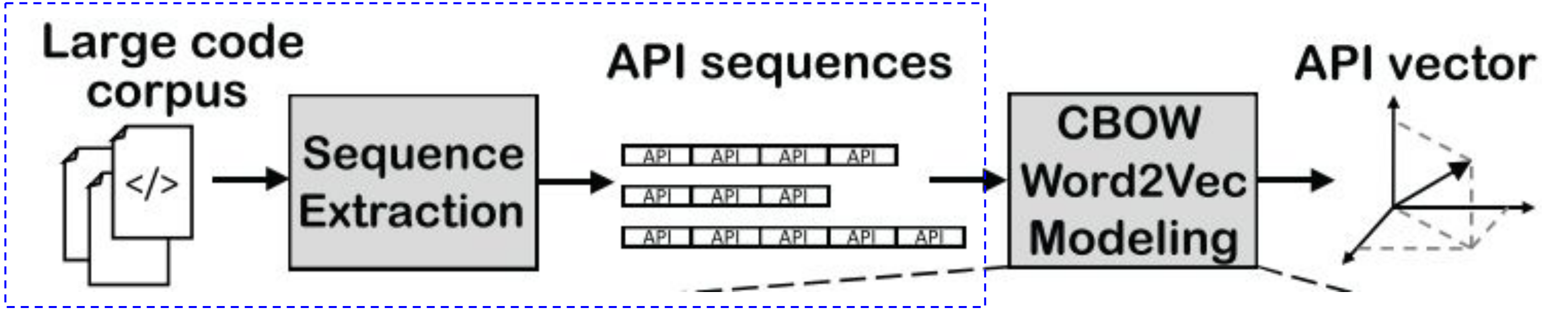
Motivation: In NLP, the regularity of words is observed as similar vector offsets between the pairs of words sharing a particular relation.

Example: "check if the current element exists before retrieval" occurs between *ListIterator.hasNext* and *ListIterator.next* and between *XMLStreamReader.isEndElement* and *XMLStreamReader.next*

Exploring API Embedding for API Usages and Applications

APPROACH

API2VEC: Building API Sequences for API Usage



Exploring API Embedding for API Usages and Applications

A
P
P
R
O
A
C
H

API2VEC: Rules to build API Sequences

Syntax	**T = typeof, RetType = return type**
Expression	
Literal: E ::= Lit	$\theta(E)$ = T(Lit) *e.g.,* θ("ABC") = String
Identifier E ::= ID	$\theta(E)$ = T(ID)#var *e.g.,* θ(writer)=FileWriter#var
MethodCall E ::= e.m(e_1,...,e_n)	$\theta(E) = \theta(e_1)$ ... $\theta(e_n)$ RetType(m)#ret $\theta(e)$#rec T(e).m T(e_1)#arg ... T(e_n)#arg Discard $\theta(e_i)$ if e_i is ID or Literal Discard θ(e)#rec if e is a class name *e.g.,* θ(dict.get(vocab)) = Integer#ret HashMap#rec HashMap.get String#arg
Constructor E ::= [e.]new C(e_1,...,e_n)	$\theta(E) = \theta(e_1)$ ... $\theta(e_n)$ [$\theta(e)$] T(C).new T(e_1)#arg...T(e_n)#arg *e.g.,* θ(new FileWriter("A")) = FileWriter.new String#arg
Field Access E::= e.f	$\theta(E)$ = T(f)#ret θ(e)#rec T(e).f Discard $\theta(e)$#rec if e is a class name *e.g.,* θ(reader.lock)=Object#ret Reader#rec Reader.lock
Variable Decl E ::= C id_1[=e_1], ...[id_n[=e_n]]	$\theta(E)$ = C#var $\theta(e_1)$ [... C#var $\theta(e_n)$] *e.g.,* θ(FileWriter writer)=FileWriter#var

Exploring API Embedding for API Usages and Applications

APPROACH

API2VEC: API Sequence Example

An API Usage in Java JDK

```
HashMap dict = new HashMap();
dict.put("A", 1);
FileWriter writer = new FileWriter("Vocabulary.txt");
for (String vocab: dict.keySet())
    writer.append(vocab + " " + dict.get(vocab)+"\r\n");
writer.close();
```

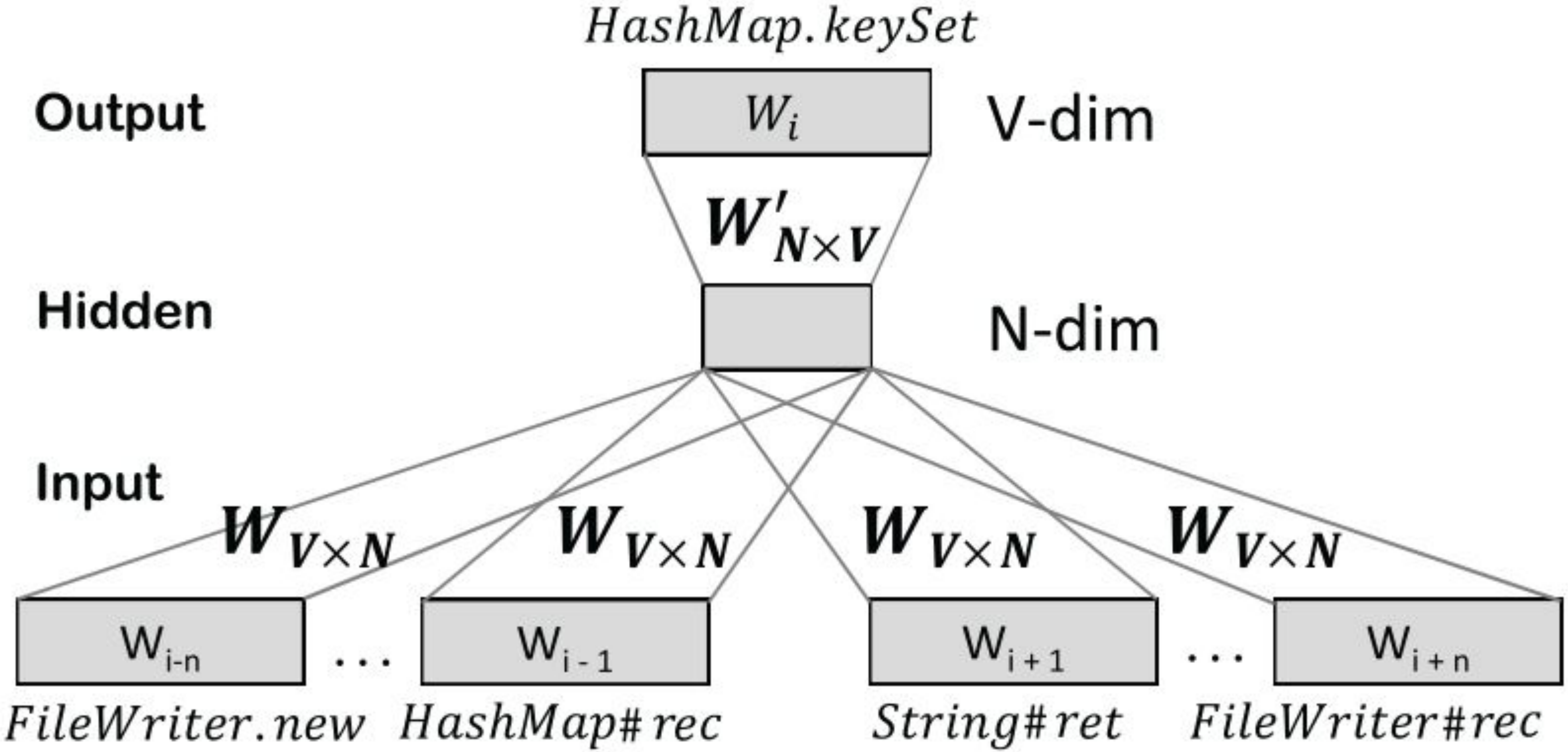
Corresponding API sequence

```
HashMap#var HashMap.new
String#ret HashMap#rec HashMap.put String#arg Integer#arg
FileWriter#var FileWriter.new String#arg
for String#var String[]#ret HashMap#rec HashMap.keySet
String#ret HashMap#rec HashMap.get String#arg FileWriter#rec
FileWriter.append String#arg
FileWriter#rec FileWriter.close
```

Exploring API Embedding for API Usages and Applications

APPROACH

API2VEC: A Training Example



Exploring API Embedding for API Usages and Applications

A
P
P
R
O
A
C
H

API2VEC: Dataset

	#projects	#Classes	#Meths	#LOCs	Voc size
Java Dataset	14,807	2.1M	7M	352M	123K
C# Dataset	7,724	900K	2.3M	292M	130K

Dataset to Build API2Vec Vectors

A
P
P
R
O
A
C
H

Exploring API Embedding for API Usages and Applications

API2VEC: Answering Research Questions

A. ***RQ1.*** *Nearby Vectors Represent APIs with Similar Contexts*

Top-5	Number	%
Similar surroundings APIs	4632	92.64
Dissimilar surroundings APIs	368	7.36

Reason
APIs have multiple contexts and some contexts with infrequently used APIs were not captured with insufficient data

G1. File.new	**G4. List.iterator**
System.getProperty ProcessBuilder.directory Path.toFile FileDialog.getFile JarFile.new	SynchronousQueue.iterator ArrayList.iterator ArrayDeque.iterator Collection.iterator Vector.iterator
G2. System.currentTimeMillis	**G5. String.hashCode**
Calendar.getTimeInMillis ThreadMXBean.getThreadUserTime Thread.sleep File.setLastModified Calendar.setTimeInMillis	Integer.hashCode Date.hashCode Class.hashCode Boolean.hashCode Long.hashCode
G3. String.compareTo	**G6. Map.keySet**
Integer.compareTo Comparable.getClass Boolean.compareTo Long.compareTo Comparable.toString	IdentityHashMap.entrySet EnumMap.entrySet AbstractMap.keySet NavigableMap.keySet IdentityHashMap.keySet

Examples of APIs sharing similar surrounding APIs

Exploring API Embedding for API Usages and Applications

A
P
P
R
O
A
C
H

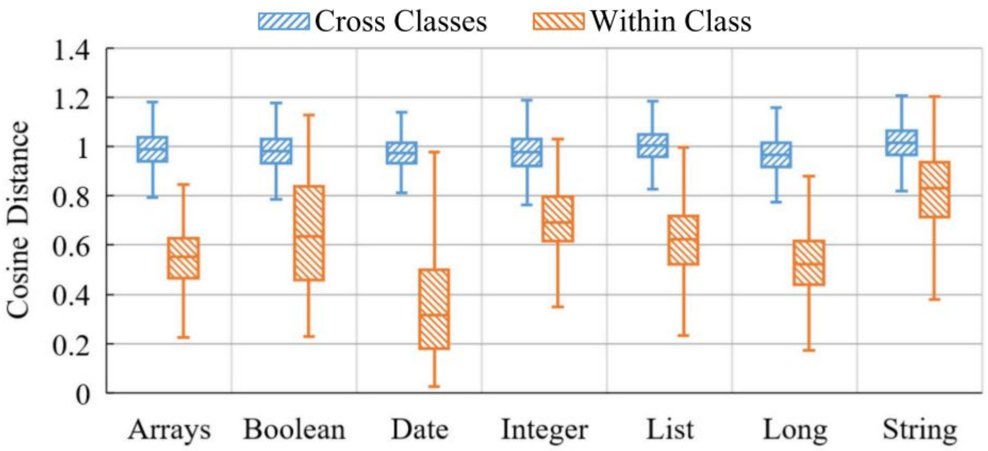
API2VEC: Answering Research Questions

***B.** An API method call or field access to be projected closer to the other APIs of the same class than the APIs of different classes*

T-test Hypothesis

Alternative Hypothesis
the distances among the vectors of the APIs within a class are smaller than the distances among the vectors of APIs belong to different classes

Null Hypothesis
those distances are equal



Distances of JDK API Vectors within and cross Classes

	t	df	p-value	Confidence interval
Java Class	-934.33	223.330	$<2.2\text{x}10^{-15}$	$(-\infty; -0.5280486)$
Java Package	-109.52	67.360	$<2.2\text{x}10^{-15}$	$(-\infty; -0.0472560)$
C# Class	-962.47	351.961	$<2.2\text{x}10^{-15}$	$(-\infty; -0.6252377)$
C# Package	-443.71	282.878	$<2.2\text{x}10^{-15}$	$(-\infty; -0.1364794)$

Exploring API Embedding for API Usages and Applications

A
P
P
R
O
A
C
H

API2VEC: Answering Research Questions

RQ2. *Similar Vector Offsets Reflect Similar Relations*

Example of vector offset

V(List.add) - V(List#var) = V(Map.put) - V(Map#var)

Candidate list	Accuracy (%)
Top - 1	74.1
Top - 5	94.2

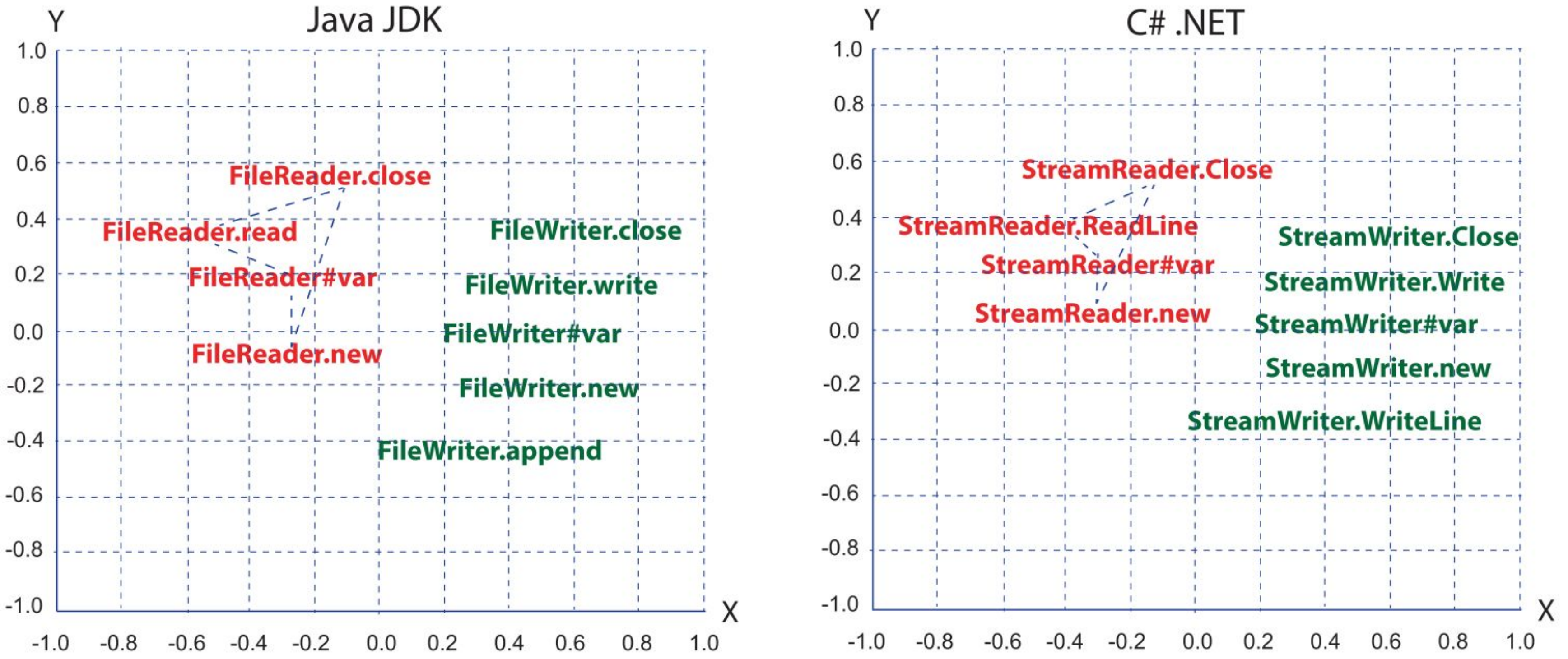
		Rank
R1. Check the current element before retrieval		
ListIterator.hasNext	ListIterator.next	1
Enumeration.hasMoreElements	Enumeration.nextElement	1
StringTokenizer.hasMoreTokens	StringTokenizer.nextToken	3
XMLStreamReader.isEndElement	XMLStreamReader.next	1
R2. Obtain property after creating system/stream		
System#var	System.getProperty	1
Properties#var	Properties.getProperty	1
XMLStreamReader#var	XML...Reader.getAttr...Value	1
R3. Add an element to various types of collections		
List#var	List.add	1
Map#var	Map.put	1
Hashtable#var	Hashtable.put	1
Dictionary#var	Dictionary.put	1
R4. Parse a string into different types of numbers		
Float#var	Float.parseFloat	1
Double#var	Double.parseDouble	1
Integer#var	Integer.parseInt	1
Long#var	Long.parseLong	1
R5. Avoid adding duplicate element to a collection		
Set.contains	Set.add	1
Map.containsKey	Map.put	3
LinkedList.contains	LinkedList.add	1
Hashtable.containsKey	Hashtable.put	3

Example Relations Via Vector Offsets in JDK

Exploring API Embedding for API Usages and Applications

APPROACH

API2API: Motivation

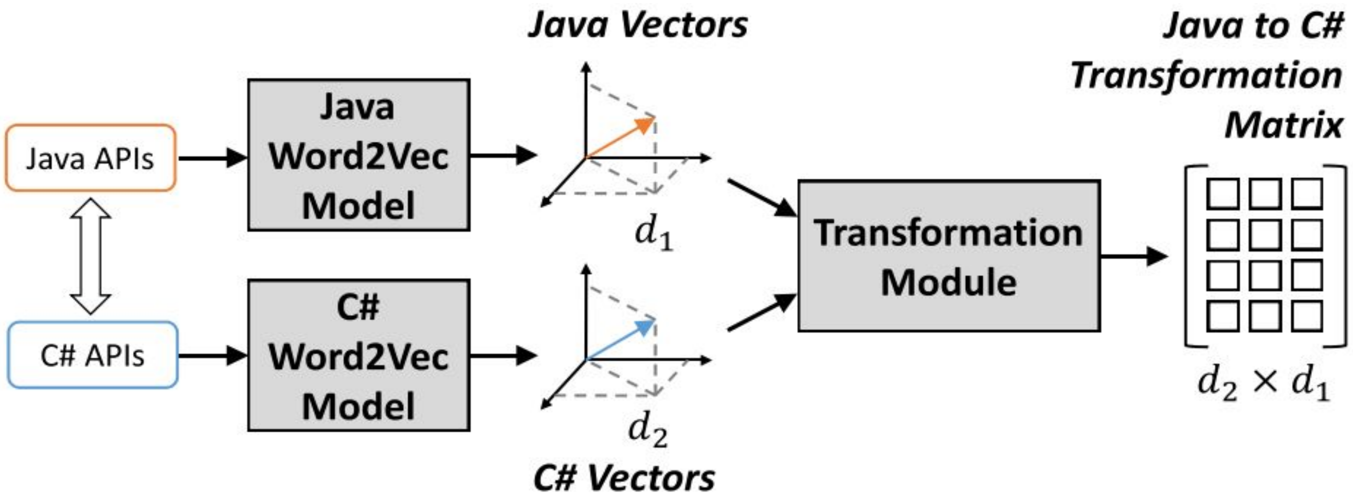


Distributed Vector Representation reduced to two dimensions using PCA for some APIs in Java and the corresponding APIs in C#

Exploring API Embedding for API Usages and Applications

A
P
P
R
O
A
C
H

API2API: Transformation Matrix



Training for the Transformation Matrix

Loss Function:

Transformation Matrix

C# Vector

$$\min_{W} \sum_{i=1}^{n} ||T \times j_i - c_i||^2$$

Java Vector

R E S U L T S

Exploring API Embedding for API Usages and Applications

API2API: Accuracy Comparison (Quantitative)

Comparison in Top-k API Mapping Mining Accuracy

*Parameters: 2*n = 10, N = 300*

R E S U L T S

Exploring API Embedding for API Usages and Applications

API2API: Accuracy Comparison (Qualitative)

1. StaMiner requires a parallel corpus of corresponding usages in two languages
2. Both the tools have out-of vocabulary issue
3. StaMiner has a stronger requirement that the mapped APIs must be in respective pairs in the parallel corpus
4. Using transformation, API2API does not need a parallel corpus with respective API usages but requires a training dataset of single API pairs
5. API2VEC need high volume of code to build high-quality vectors

API2API: Accuracy Comparison (New API Mappings)

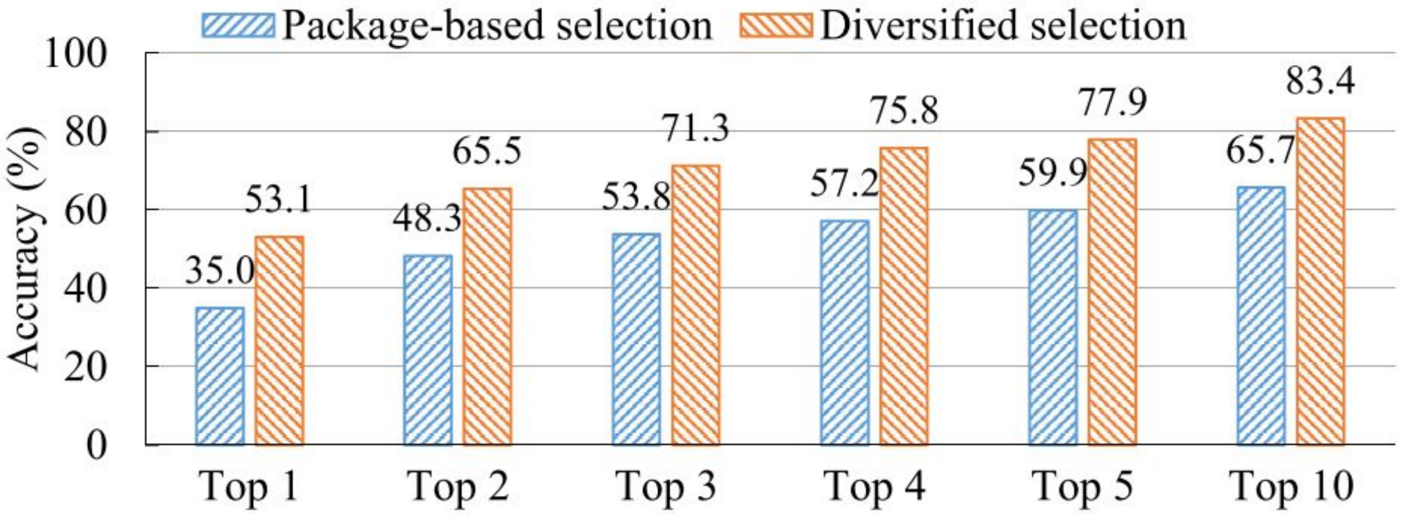
Technique	Number of new API Mappings found
StaMiner [11]	25
API2API	52

Exploring API Embedding for API Usages and Applications

RESULTS

API2API: Ablation Study

1. Selecting different packages of API mapping pairs to train the transformation matrix



Top-k Accuracy with different Training Data Selection

Package-based selection: Divided in 13 groups, trained and tested in groups

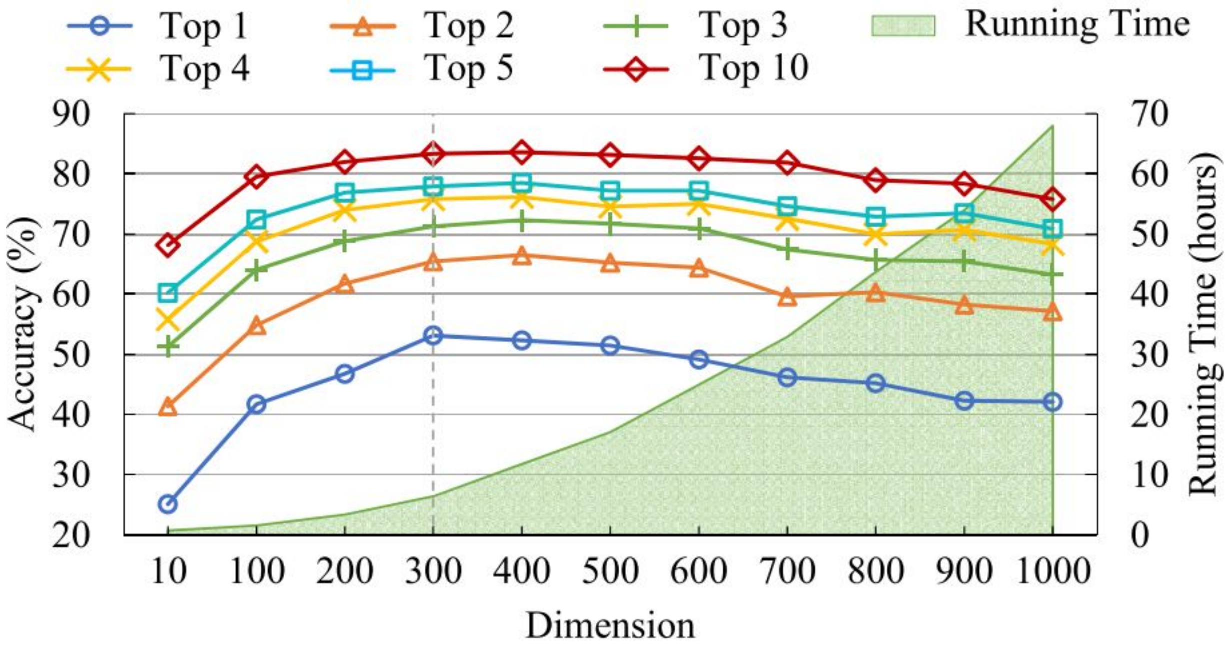
Diversified selection: 10-fold cross validation with random selection from each package

R
E
S
U
L
T
S

Exploring API Embedding for API Usages and Applications

API2API: Ablation Study

2. Varying Numbers of Dimensions of Vector Spaces



Top-k Accuracy with different Numbers of Dimensions

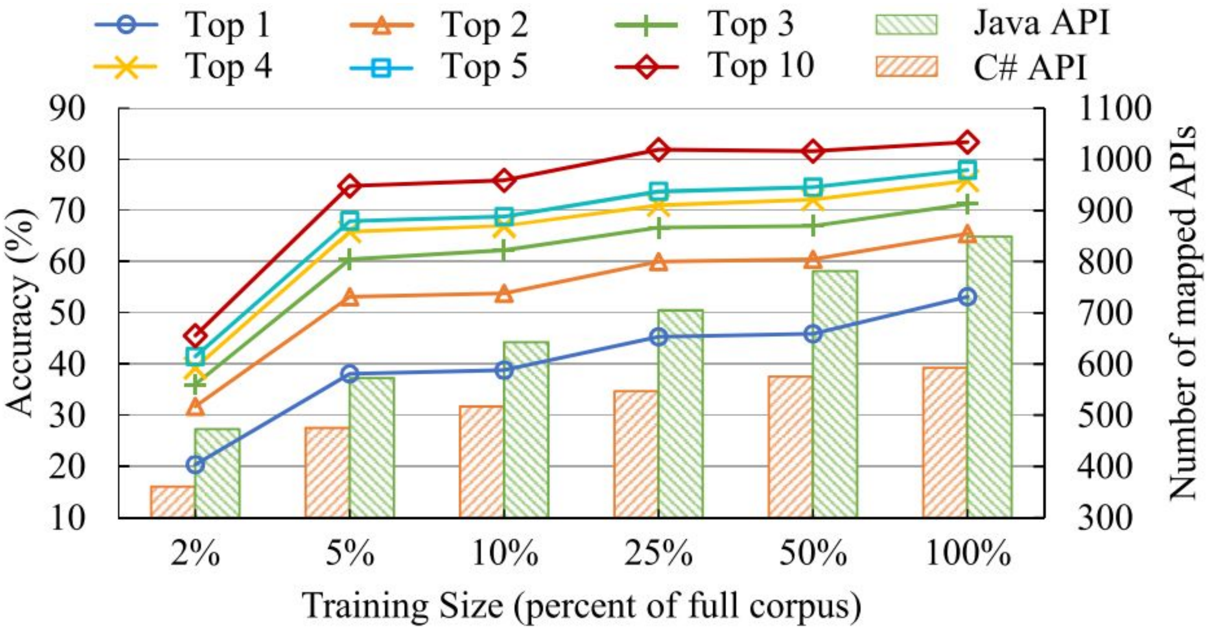
Exploring API Embedding for API Usages and Applications

R
E
S
U
L
T
S

API2API: Ablation Study

3. Varying Word2Vec Window's Sizes: Optimal results for 10

4. Varying Sizes of Training Datasets for Word2Vec



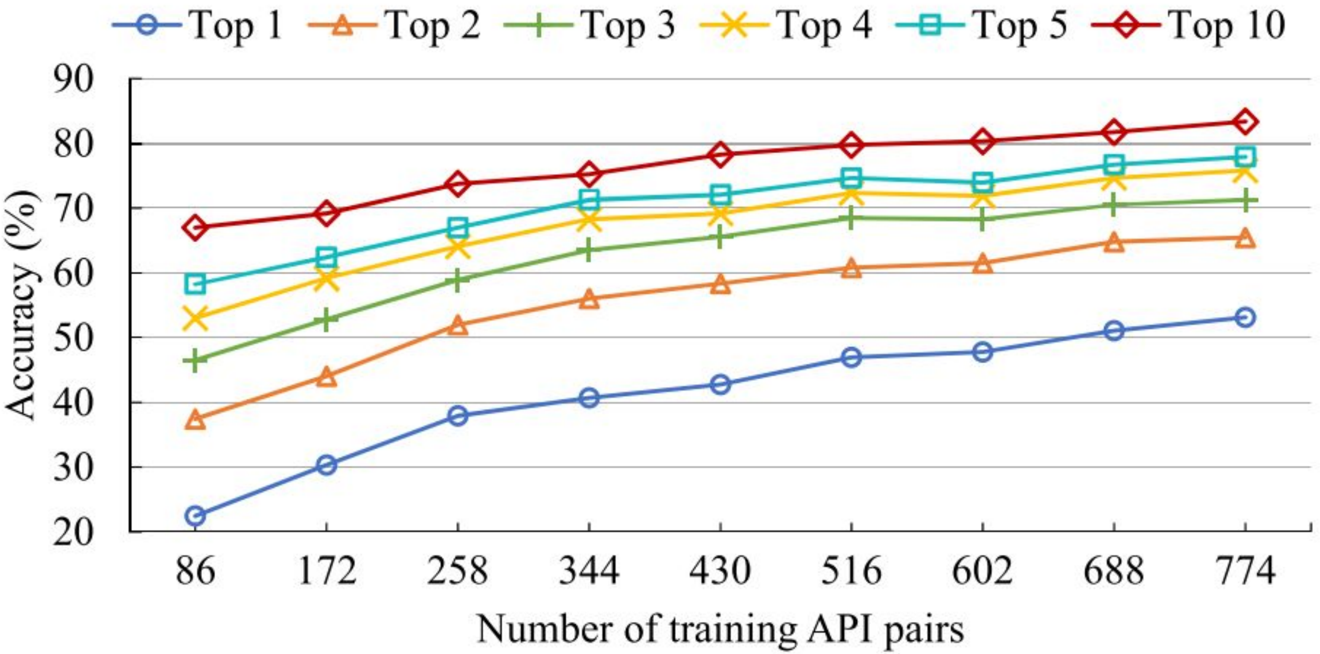
Accuracy with Varied Training Datasets for Word2Vec

Exploring API Embedding for API Usages and Applications

R E S U L T S

API2API: Ablation Study

5. Varying number of mapping pairs to train the transformation matrix

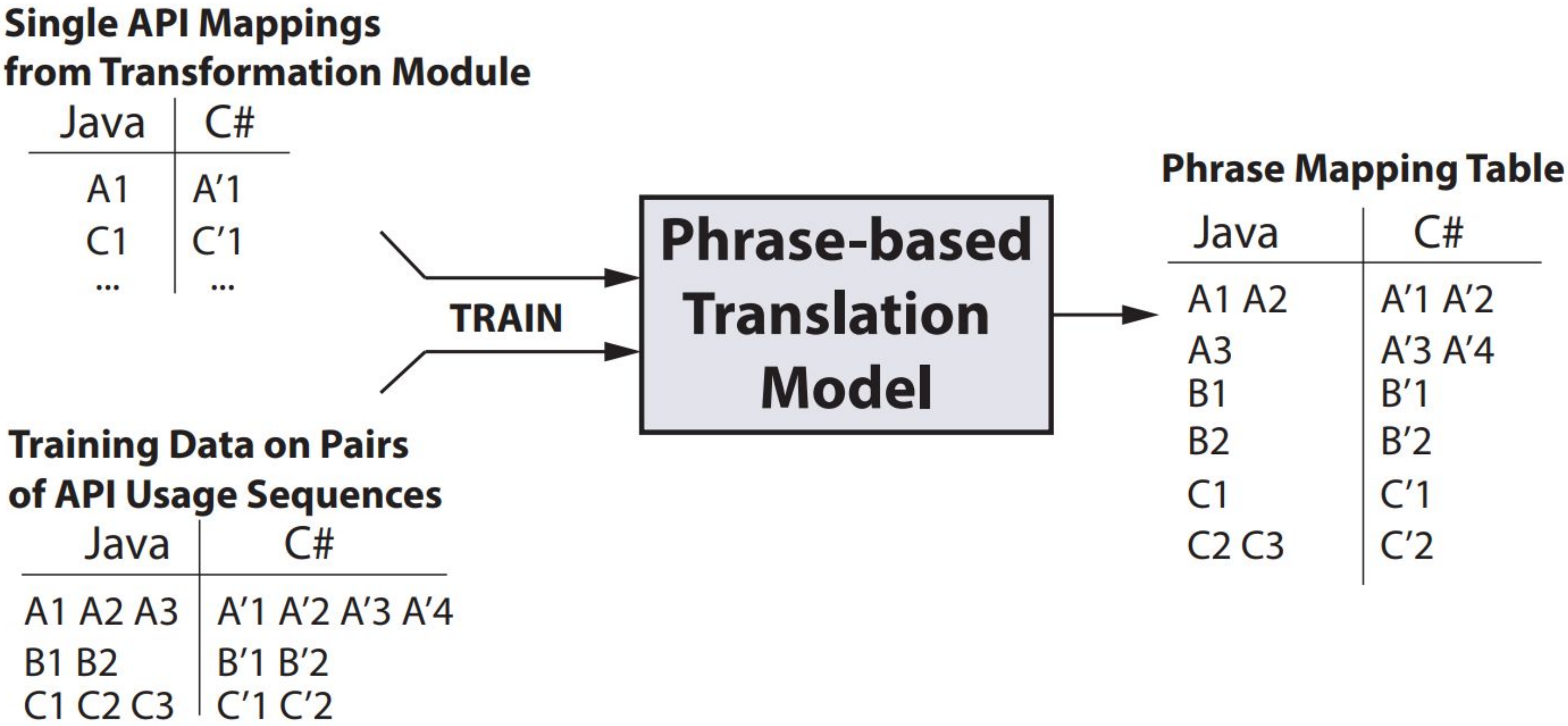


Accuracy with various Numbers of Training Mappings

APPLICATION

Exploring API Embedding for API Usages and Applications

MIGRATING EQUIVALENT API USAGE SEQUENCES



Training for Phrase-based Translation Model

A P P L I C A T I O N

Exploring API Embedding for API Usages and Applications

Example: MIGRATING EQUIVALENT API USAGE SEQUENCES

Input source code in Java →

```
HashMap dict = new HashMap();
dict .put("A", 1);
FileWriter writer = new FileWriter("Vocabulary.txt");
for (String vocab: dict.keySet()){
  writer .append(vocab + " " + dict.get(vocab)+"\r\n");
}
writer .close();
```

Translate to C#

Extract sequence of APIs in Java →

```
HashMap.var
HashMap.new
HashMap.put
FileWriter.var
FileWriter.new
String
for
String.var
HashMap.keySet
FileWriter.append
String
HashMap.get
String
FileWriter.close
```

Generated sequence of APIs in C# →

```
Dictionary.var
Dictionary.new
Dictionary.Add
StreamWriter.var
StreamWriter.new
String
Foreach
String.var
Dictionary.Keys
int.var
Dictionary.TryGetValue
String
int
StreamWriter.WriteLine
String
StreamWriter.Close
```

A
P
P
L
I
C
A
T
I
O
N

Exploring API Embedding for API Usages and Applications

MIGRATING EQUIVALENT API USAGE SEQUENCES

Dataset: Oracle O

Users partially migrated a project

Migrate a new project fully

Project	Within-Project		Cross-Project	
	Recall	Precision	Recall	Precision
Antlr	87.8	75.2	**90.6**	**87.2**
db4o	83.9	79.4	88.7	75.8
Fpml	**89.6**	**86.1**	86.3	83.7
Itext	75.9	77.2	76.5	81.3
JGit	77.2	**66.4**	81.1	**67.1**
JTS	76.3	76.6	76.3	73.7
Lucene	**75.7**	77.7	**77.1**	78.5
Neodatis	78.6	70.4	78.8	74.2
POI	76.9	78.3	77.1	78.6

Accuracy (%) In Generating Equivalent API Usage Sequences

Precision: LCS / Result

Recall: LCS / Reference

LCS: Longest Common subsequence

R E L A T E D W O R K

Exploring API Embedding for API Usages and Applications

Related Work

1. DeepAPI [2] uses Recurrent Neural Network Enc./Dec. to generate API sequences (translational model between texts and API sequences)

2. Ye et al. [3] used Skip-gram model on API , reference documents and tutorials to create embeddings. Aimed to quantify relations between words and elements to improve text code retrieval.

3. PAM [4] (parameter-free probabilistic algorithm to mine API patterns)

4. Allamanis et al. [5] suggests methods/classes name using embeddings(statistical coocurrences projected into continuous space with words from the names

5. Maddison et al.[6] and Anycode[7] use probabilistic CFGs and neuro-probabilistic language models code.

6. Allamanis et al.[8] use bimodel modeling for short texts and source code snippets.

7. **API2API is inspired from Mikolov et al.[9] where similar geometric arrangements were observed in English and Spanish words for numbers and animals.**

8. Mou et al.[10] use convolutional neural networks over tree structures which can be replaced inplace of Word2Vec in API2API

9. StaMiner [11] mines API mappings by maximizing the likelihoods of observing the mappings between API pairs from a parallel corpus of client code.

D I S C U S S I O N

Exploring API Embedding for API Usages and Applications

Questions and Discussions

1. Dataset (Lots of it !!)
2. Similar mappings might not be observed on other programming languages pair like python and C++ (Static and Dynamic Compile time languages)
3. Only works for one to one mapping, what about n-to-1 or 1-to-n mappings. What could be some potential solutions?
4. Stochastic Gradient Descent suffers from some limitations like local minima and saddle point problems. Other strategies that can be explored are Adaptive Moment Estimation(ADAM) and Nestrov Accelerated Gradient.
5. How to generalize the approach. There is a possibility that the approach works well only for the dataset they consider and do not generalize well as is the case with most machine learning strategies.
6. What can be the probable strategies of handling previously unseen APIs ?

Exploring API Embedding for API Usages and Applications

R E F E R E N C E S

References

[1] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013 (NIPS'13)., 2013, pp. 3111–3119.

[2] X. Gu, H. Zhang, D. Zhang, and S. Kim, "Deep API learning," in Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ser. FSE 2016. New York, NY, USA: ACM, 2016, pp. 631–642. [Online]. Available: http://doi.acm.org/10.1145/2950290.2950334.

[3] X. Ye, H. Shen, X. Ma, R. Bunescu, and C. Liu, "From word embeddings to document similarities for improved information retrieval in software engineering," in Proceedings of the 38th International Conference on Software Engineering, ser. ICSE '16. ACM, 2016, pp. 404–415. [Online]. Available: http://doi.acm.org/10.1145/2884781.2884862.

[4] J. M. Fowkes and C. A. Sutton, "Parameter-free probabilistic API mining at GitHub scale," CoRR, vol. abs/1512.05558, 2015. [Online]. Available: http://arxiv.org/abs/1512.05558.

[5] M. Allamanis, E. T. Barr, C. Bird, and C. Sutton, "Suggesting accurate method and class names," in Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ser. ESEC/FSE 2015. New York, NY, USA: ACM, 2015, pp. 38–49. [Online]. Available: http://doi.acm.org/10.1145/2786805.2786849.

[6] C. J. Maddison and D. Tarlow, "Structured generative models of natural source code," in The 31st International Conference on Machine Learning (ICML), June 2014.

[7] T. Gvero and V. Kuncak, "Synthesizing Java expressions from free-form queries," in Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications, ser. OOPSLA 2015. New York, NY, USA: ACM, 2015, pp. 416–432. [Online]. Available: http: //doi.acm.org/10.1145/2814270.2814295.

[8] M. Allamanis, E. T. Barr, C. Bird, and C. Sutton, "Learning natural coding conventions," in Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ser. FSE 2014. New York, NY, USA: ACM, 2014, pp. 281–293. [Online]. Available: http://doi.acm.org/10.1145/2635868.2635883.

[9] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation." CoRR, vol. abs/1309.4168, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1309. html#MikolovLS13.

[10] H. Peng, L. Mou, G. Li, Y. Liu, L. Zhang, and Z. Jin, "Building program vector representations for deep learning," Knowledge Science, Engineering and Management, Lecture Notes in Computer Science, vol. 9403, pp. 547–553, 2015.

[11] A. T. Nguyen, H. A. Nguyen, T. T. Nguyen, and T. N. Nguyen, "Statistical Learning Approach for Mining API Usage Mappings for Code Migration," in Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, ser. ASE '14. New York, NY, USA: ACM, 2014, pp. 457–468. [Online]. Available: http://doi.acm.org/10.1145/2642937.2643010.

Exploring API Embedding for API Usages and Applications

APPENDIX