

Import2vec Learning Embeddings for Software Libraries

**Bart Theeten
Frederik Vandeputte
Tom Van Cutsem**

参考サイト（著者のサイト）

<https://www.code-compass.com/blog/intro/>

概要

- 簡単にまとめると . . .
- Word2vecを使ってライブラリをベクトル化する
 - ライブラリをクラスタリングしてカテゴリ化
 - 類似ライブラリ推薦
などに使える

背景

- 背景

- ソフトウェアライブラリの総数が増加している
 - ・ ソフトウェア開発の生産性を向上させている

- 課題

- 多くのライブラリが存在するため、開発の目的にあったライブラリを探すことが困難

- 現在の解決策

- 手動によるライブラリの索引付け
 - ・ Githubのawesome lists（カテゴリ別のライブラリリスト）
 - ・ www.libhunt.com（23,000プロジェクトの索引付け）

Big Data

- [Big Data](#)
- [Public Datasets](#)
- [Hadoop](#) - Framework for distributed storage and processing of very large data sets.
- [Data Engineering](#)
- [Streaming](#)
- [Apache Spark](#) - Unified engine for large-scale data processing.

 **Awesome Android**
android.libhunt.com

 **Awesome C++**
cpp.libhunt.com

 **Awesome Crystal**
crystal.libhunt.com

 **Awesome Elixir**
elixir.libhunt.com

 **Awesome Go**
go.libhunt.com

 **Awesome Haskell**
haskell.libhunt.com

 **Awesome iOS**
ios.libhunt.com

 **Awesome Java**
java.libhunt.com

 **Awesome JS**
js.libhunt.com

ライブラリの索引付け

- 手作業の索引付けにおける課題

- 人気のパッケージリポジトリには100万以上のライブラリが存在しており，ほとんどが索引付けされていない

- 機械学習におけるアプローチ

- 機械学習を用いることで，ライブラリを自動で索引付けできる可能性はある

➡ 機械学習を上手く適用するためには，ライブラリを数学的（ベクトル）に表現する必要がある

- 類似したライブラリは類似した表現で表せるような．．．

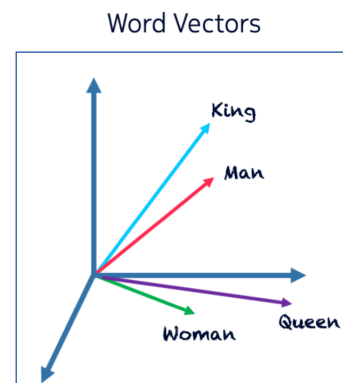
目的・アプローチ

• 目的

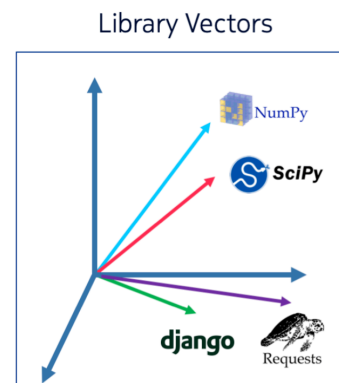
- 自然言語処理技術を用いることで、ライブラリのベクトルを表現することが可能か調査する
 - ・ 意味的な類似性を考慮したクラスタリングができれば、開発者はライブラリを探しやすくなる

• アプローチ

- Word2vecを用いることで、ライブラリのベクトル表現の取得を行なう



Conjecture: words that frequently occur in the same context will tend to be more semantically similar



Conjecture: software libraries that are frequently imported together will tend to be more compatible

ライブラリのペアを学習させることで
Library Vectorsを取得できるのでは？

ライブラリの意味的表現

- 類似ライブラリの定義

- APIドキュメントの内容が似ている（同じ機能を持つ）
- 一緒に利用されるライブラリ
 - numpyとscipy（scipyではnumpyで使われる型を計算する）
- Word2vecを利用することで共起パターンを捉えることができると思われる

```
import numpy as np
from scipy import linalg
A = np.array([[1,2],[3,4]])
linalg.inv(A) # invert matrix A
```

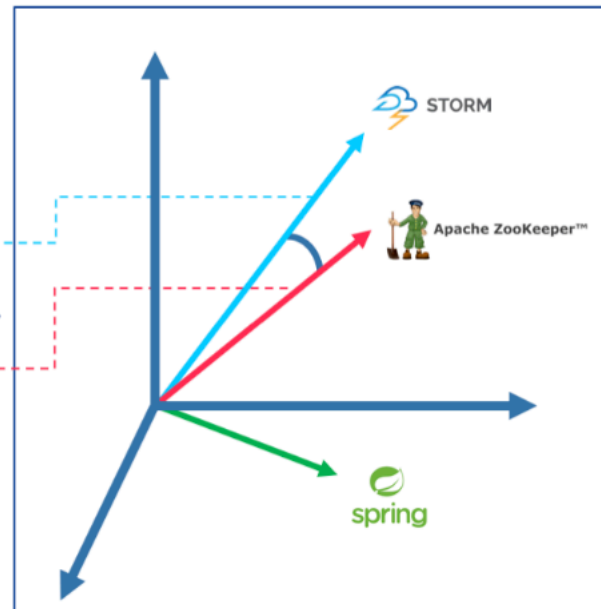
numpyとscipyの組み合わせが複数ファイルで
出現するとライブラリ間で密接な関係があると思われる

アプローチの概要

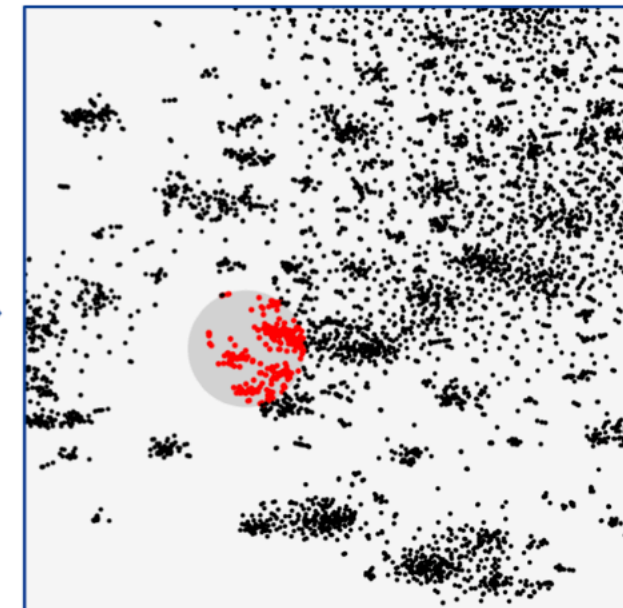
Extract library imports
from source code

```
1 package backtype.storm.transactional.state;
2
3 import backtype.storm.Config;
4 import backtype.storm.serialization.KryoValuesDeserializer;
5 import backtype.storm.serialization.KryoValuesSerializer;
6 import backtype.storm.utils.Utils;
7 import com.netflix.curator.framework.CuratorFramework;
8 import java.util.ArrayList;
9 import java.util.HashMap;
10 import java.util.List;
11 import java.util.Map;
12 import org.apache.zookeeper.CreateMode;
13 import org.apache.zookeeper.KeeperException;
14
15 public class TransactionalState {
16     ...
17 }
```

Learn vector representation



Compute nearest neighbors



importライブラリの
抽出

Skip-gramモデルを
用いて学習

ベクトル間の
距離を計算

Lang.	Extracted import statements
Java	<code>import X.Y[.*];</code>
JS	<code>require('X') & import Y[,...] from X</code>
Python	<code>import X.Y[.*] & from X.Y import Z</code>
Ruby	<code>require[_relative] 'X'</code>
PHP	<code>use [function const] X\Y[\{Z,...}]</code>
C#	<code>using [static] X.Y[.*];</code>

ライブラリを抽出するための
正規表現

抽出ライブラリの例

- 1ソースファイルに含まれるライブラリの組み合わせを1セット
 - ライブラリの最上位パッケージが対象
 - **os.path**, **data.base_dataset.BaseDataset**, **PIL.Image**

TABLE III: Raw imports extracted from 5 representative source files of the pytorch CycleGAN-and-pix2pix GitHub project.

Source file	Imports
data/single_dataset.py	os.path, data.base_dataset.BaseDataset, data.base_dataset.get_transform, data.image_folder.make_dataset, PIL.Image
models/cycle_gan_model.py	torch, itertools, util.image_pool.ImagePool, base_model.BaseModel
models/networks.py	torch, torch.nn, torch.nn.init, functools, torch.optim.lr_scheduler
util/get_data.py	__future__.print_function, os, tarfile, requests, warnings.warn, zipfile.ZipFile, bs4.BeautifulSoup, os.path.abspath, os.path.isdir, os.path.join, os.path.basename
util/visualizer.py	numpy, os, sys, ntpath, time, scipy.misc.imresize

Skip-gram Model

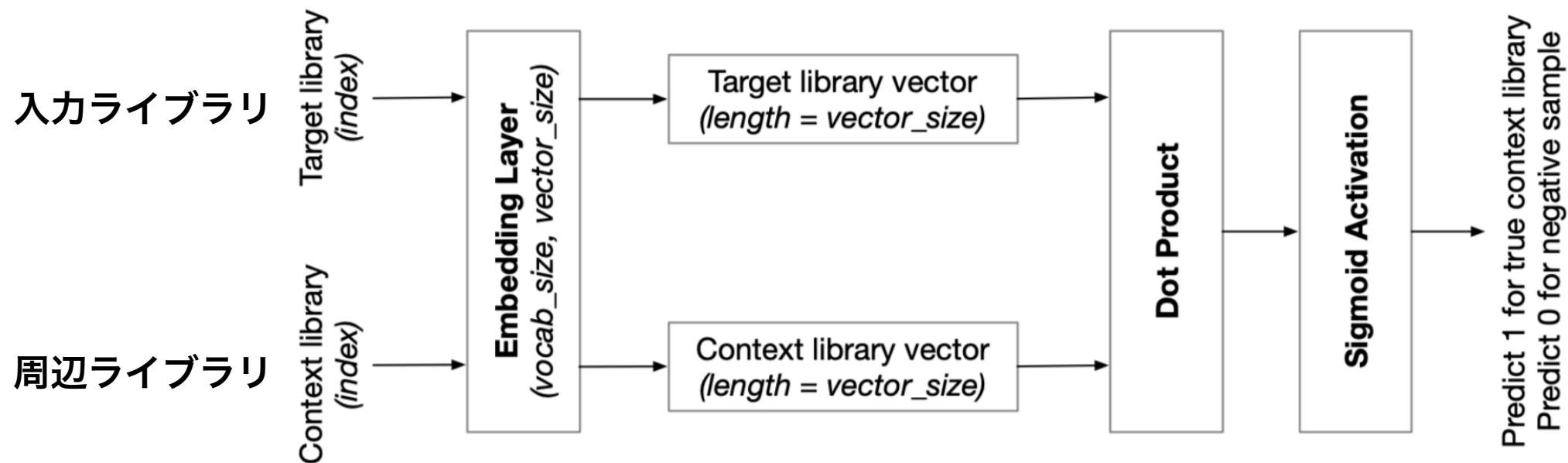


Fig. 11: The Skip-gram Model to Train Import Vectors

- 例

- util/visualizer.py
[numpy, os, sys, ntpath, time, scipy]

Target library : [numpy]

Context library : [os, sys, ntpath, time, scipy]

対象プロジェクト

TABLE II: Basic statistics of all crawled open source projects

Language Repo	Java		JS		Python		Ruby		PHP		CSharp	
	MVNCentral	GitHub	NPM	GitHub	PyPi	GitHub	RubyGems	GitHub	Packagist	GitHub	Nuget	GitHub
# Projects	120K	260K	380K	310K	110K	260K	130K	110K	130K	69K	100K	93K
# Removed Clones	3%	2.4%	19%	9.2%	9.7%	5.8%	5.5%	12%	12%	8.6%	8.7%	3.2%
# Source Files	5M	24M	3.8M	6.5M	1.5M	7.6M	1.3M	2.7M	2.7M	4.8M	5M	8.3M
# Import Stmts	38M	200M	16M	34M	9.3M	51M	3.1M	5.8M	8.6M	15M	17M	37M
# Unique Imports	2M	8.6M	1.4M	2.5M	1.6M	3.6M	690K	570K	1.1M	870K	310K	540K

対象プロジェクト

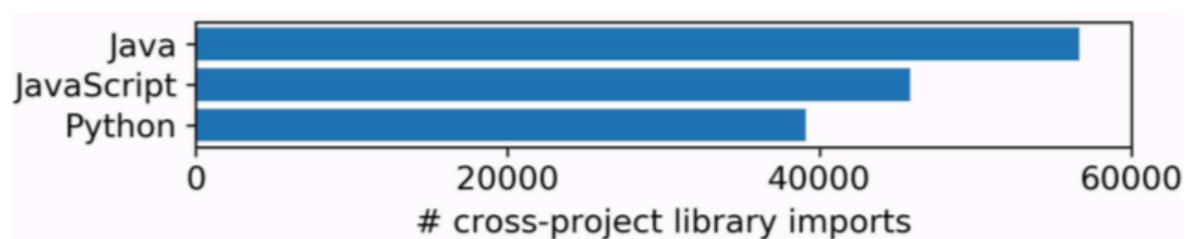


Fig. 8: # unique library imports used for training embeddings.

ライブラリ数

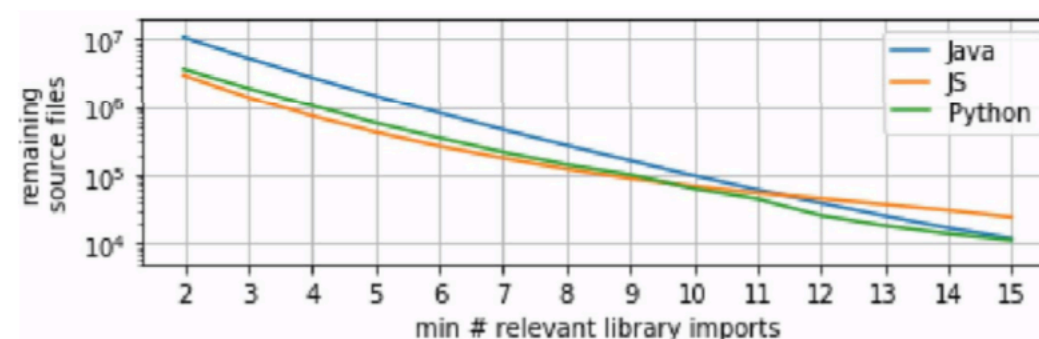


Fig. 9: # source files with at least R relevant library imports.

共起回数

手法を適用した結果の分析

1.作成したモデルの定性的分析

- t-SNEによる可視化

2.Word2vecの結果と共起回数をカウントした結果との比較

3.プロジェクト間の比較

- 類似プロジェクトでどれだけ同じライブラリがimportされているか

作成したモデルの定性的分析

- モデルのt-SNEによる可視化
 - Word2vecとかの定性的評価のために使われる可視化手法

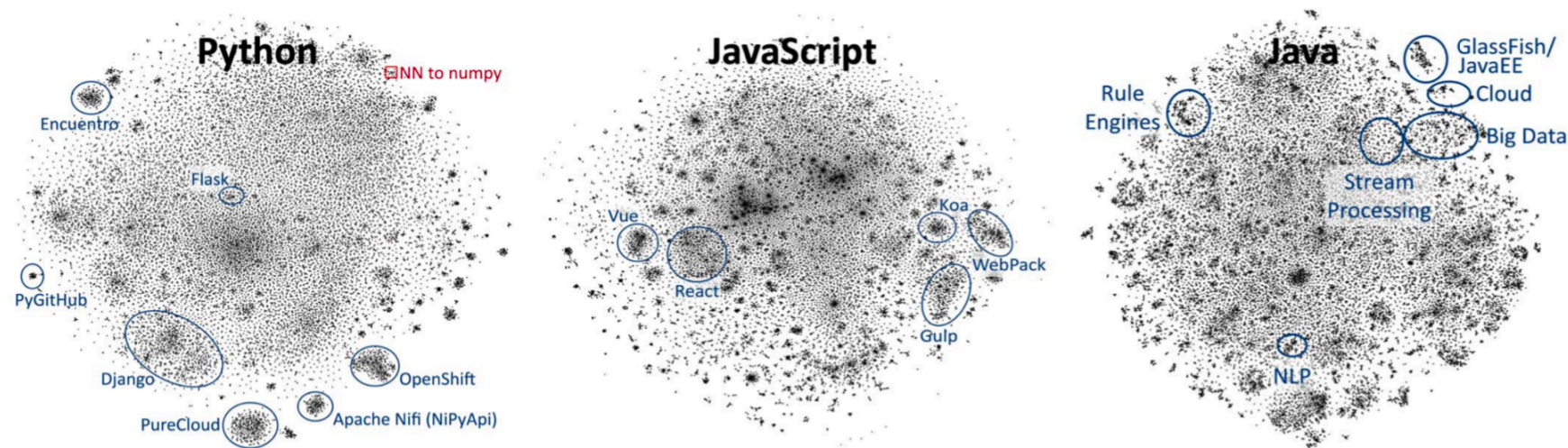


Fig. 12: 2D-visualization of the Model output for the Python, JavaScript and Java ecosystems

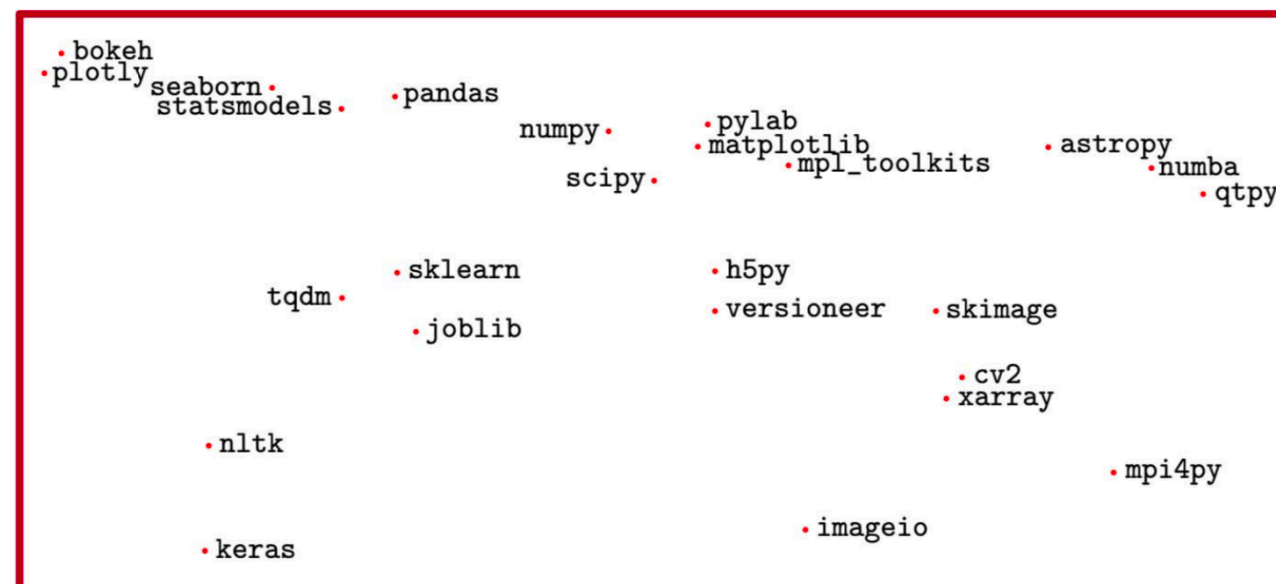


Fig. 13: A zoom on the local neighborhood of numpy (Python)

Word2vecと共起回数との比較

- Library(n) : nはライブラリのimport回数の順位

TABLE V: Top-5 Nearest Neighbours vs top-5 most frequently co-occurring libraries for some well-known given libraries

	Given Library	Top-5 Nearest Neighbours	Top-5 Most Popular combos
Java	org.apache.spark	co.cask.cdap:cdap-spark-core (1196), ch.cern.hadoop (114), com.facebook.presto.hadoop (247), com.moz.fiji.mapreduce (381), com.clearspring.analytics (1650)	com.h3xstream.findsebugs (7), ch.cern.hadoop (114), com.google.guave (16), org.slf4j (8), junit (73)
	org.apache.lucene	org.hibernate:search-analyzers (821), org.elasticsearch (231), nexus.bundles.elasticsearch (836), org.jboss.seam.embedded (2451), org.apache.jackrabbit:oak-lucene (4710)	org.jboss.seam.embedded (2451), junit (73), org.infinispan (581), org.apache.solr (948), org.elasticsearch (231)
	org.apache.opennlp	com.graphaware.neo4j:nlp (10393), org.apache.uima (522), org.apache.tika (2003), org.apache.ctakes (3555), edu.ucar.netcdf (2257)	junit (73), java-nio-charset (13), java-util-regexp (9), org.apache.uima (522), org.slf4j (8)
	org.apache.maven	org.apache.maven.plugin-tools (120), org.codehaus.plexus (109), org.eclipse.aether (414), org.sonatype.plexus (851), org.apache.maven.shared (1274)	org.codehaus.plexus (109), org.apache.maven.plugin-tools (120), junit (73), org.eclipse.equinox (2774), commons-io (34)
JS	http	https (31), url (14), express (3), request (17), querystring (32)	assert (10), https (31), url(14), express (3), net (48)
	mysql	pg (299), redis (117), knex (343), mongodb (97), nodemailer (153)	express (3), body-parser (13), async (25), lodash (12), request (17)
	jquery	bootstrap (762), moment (27), underscore (22), angular (183), d3 (226)	moment (27), datatables (520), underscore (22), backbone (272), lodash (12)
	gm (GraphicsMagick)	imagemagick (1517), sharp (1040), connect-busboy (1913), jimp (1010), canvas (350)	async (25), request (17), express (3), lodash (12), crypto (16)
Python	scipy	numpy (23), matplotlib (58), sklearn (123), mpl_toolkits (219), pylab (228)	numpy (23), matplotlib (58), warnings (32), sklearn (123), math (28)
	tensorflow	keras (349), torch (476), cv2 (259), google (162), sklearn (123)	numpy (23), time (8), object_detection (8850), six (30), collections (11)
	http	ssl (120), httplib (142), socketserver (276), socket (34), cookielib (376)	urllib (19), httplib (142), io (18), urllib2 (65), urlparse (51)
	asyncio	aiohttp (167), async_timeout (574), uvloop (782), concurrent (147), websockets (625)	logging (6), time (8), aiohttp (167), datetime (10), functools (15)

Word2vecでは共起回数と比較すると， 順位の低いライブラリが出現する

プロジェクト間の比較

1. プロジェクトベクトル V_p を定義（各プロジェクトをベクトル化）

$$V_P = \{v_p | p \in D_t, v_p = \sum_{l \in I_p} v_l / |I_p|\}$$

v_l : ライブラリベクトル, I_p : 含まれるライブラリの数

2. 各言語から、ランダムで1,000プロジェクト抽出し

類似プロジェクトに同じライブラリが含まれるか調査

- 類似したプロジェクトには同様のライブラリが含まれていないはず

プロジェクト間の比較

- ランダムで選択した1000プロジェクトで実験
類似プロジェクト間で同じライブラリを使われている割合を算出
 - k : 対象とする類似プロジェクト数（距離が近いk個のプロジェクト）
 - context size : 比較するライブラリの数
 - 対象プロジェクトでのimport回数の多いライブラリ上位

TABLE VI: Precision of evaluation algorithm. Left: considering only 1 NN project, Right: considering 20 NN projects

context size	$k = 1$ NN project			$k = 20$ NN projects		
	2	5	10	2	5	10
Java	0.84	0.80	0.70	0.85	0.71	0.49
JavaScript	0.75	0.56	0.47	0.71	0.45	0.32
Python	0.74	0.56	0.43	0.86	0.58	0.38

Word2vecで
プロジェクトベクトルを計算

TABLE VII: Precision when using random library vectors

context size	$k = 1$ NN project			$k = 20$ NN projects		
	2	5	10	2	5	10
Java	0.55	0.56	0.56	0.58	0.60	0.41
JavaScript	0.45	0.37	0.37	0.55	0.40	0.26
Python	0.68	0.46	0.33	0.80	0.55	0.33

ランダムで
プロジェクトベクトルを割当

ランダムよりもライブラリベクトルを正しく捉えている

まとめ・所感

- まとめ

- Word2vecにライブラリのセットを学習させることで、ライブラリベクトルを獲得する手法を提案
- ライブラリベクトルの評価実験を行った

- 所感

- 評価の方法が結構無理やり？
- Word2vecを使うメリットをもう少し説明してほしかった
 - ・ Word2vecのメリットがライブラリをベクトル化できるからクラスタリングとかできる？
- API2VECとの違いは？（似てるはずやけど、引用されてなかった）
 - ・ T. D. Nguyen, A. T. Nguyen, H. D. Phan and T. N. Nguyen, "Exploring API Embedding for API Usages and Applications," 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), Buenos Aires, 2017, pp. 438-449.