

Cross-language clone detection by learning over abstract syntax trees

Daniel Perez, Shigeru Chiba

MSR '19 Proceedings of the 16th International Conference
on Mining Software Repositories Pages 518-528

背景

- 複数の言語を用いて開発されることがある
- 複数の言語で書かれることで異なるサービス間でコードと機能の重複を追跡することが困難
- 異なる言語間でコードと機能の重複を自動的に検出する技術が開発されていない

目的・アプローチ

- 目的
 - 異なる言語間でコードクローンを検出する手法を構築する
- アプローチ
 - ソースコードから訓練用データの生成
 - skip-gramモデルの訓練してクローン検出モデルを構築
 - クローン検出モデルの性能評価

訓練用データの生成

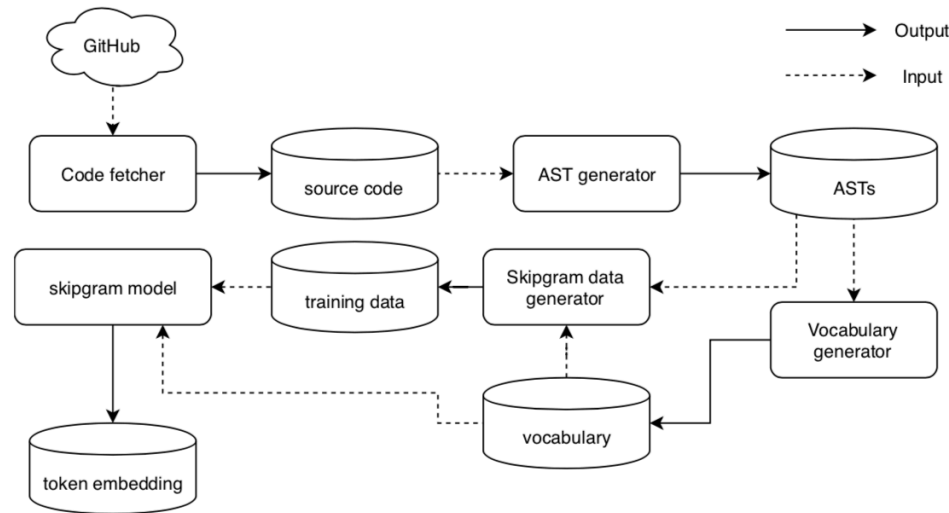


Fig. 3. Token-level vectors generation overview

- ASTの情報からボキャブラリを生成
- ASTとボキャブラリと元に字句レベルのベクトルを生成

ボキャブラリ生成

- skip-gramモデル訓練時に必要となるボキャブラリを生成
- 入力
 - ファイル, 変数の値, ボキャブラリの大きさの最大値
- 出力
 - ボキャブラリ

Algorithm 1 Vocabulary generation algorithm

```
1: function GENERATEVOCABULARY(files, includeValues, maxSize)
2:   tokensCount  $\leftarrow$  empty map
3:   for file in files do
4:     ast  $\leftarrow$  generate_ast(file)
5:     for token in ast do
6:       if (token.type, null)  $\notin$  tokensCount then
7:         tokensCount[(token.type, null)]  $\leftarrow$  0
8:       Increment tokensCount[(token.type, null)]
9:       if includeValues  $\wedge$  token.value  $\neq$  null then
10:        if (token.type, token.value)  $\notin$  tokensCount then
11:          tokensCount[(token.type, token.value)]  $\leftarrow$  0
12:        Increment tokensCount[(token.type, token.value)]
13:   tokensCount  $\leftarrow$  reverse_sort(tokensCount)
14:   vocabulary  $\leftarrow$  first maxSize keys of tokensCount
15:   return vocabulary
```

- ASTに含まれる字句それぞれが, ForStmtやIfStmtなどの言語の予約語や変数名, 関数名などの何にあたるかを判定してボキャブラリ候補に登録
- 変数の場合は型と値の組にして1つのボキャブラリ候補として登録
- それぞれの字句の出現回数を登録
- 字句を出現回数が多い順にソートして上位maxSize分をボキャブラリとする

字句レベルのベクトル生成

- 訓練用のデータとなる字句レベルのベクトル生成
- ASTの各ノードの前後関係を取得

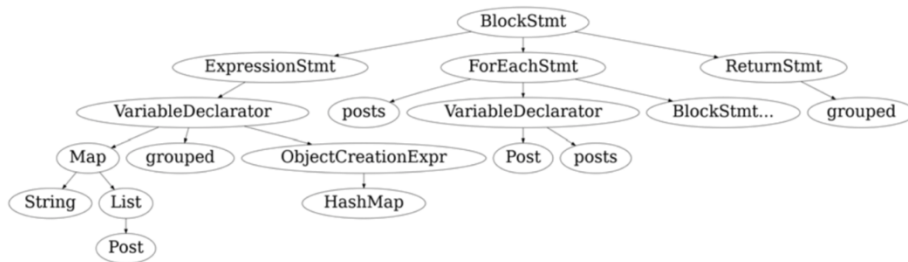


Fig. 1. Java groupPosts method AST

Algorithm 2 Data generation for skip-gram model

```
1: function GENERATESKIPGRAMDATA(files, vocabulary, params)
2:   skipgramData ← {}
3:   for file in files do
4:     ast ← GenerateAST(file)
5:     for node in ast do
6:       nodeIndex ← LookupTokenIndex(node)
7:       contextNodes ← GenerateContext(node, params)
8:       for contextNode in contextNodes do
9:         contextIndex ← LookupTokenIndex(contextNode)
10:        skipgramData.add((nodeIndex, contextIndex))
11:   return skipgramData
```

Algorithm 3 Context generation for an AST node

```
1: function GENERATECONTEXT(node, params)
2:   contextNodes ← FindDescendants(node, params.descendantWS, 0)
3:   parent ← node.parent
4:   n ← 0
5:   while parent is defined ∧ n < params.ancestorWS do
6:     contextNodes.add(parent)
7:     parent ← parent.parent
8:     n ← n + 1
9:   if params.includeSiblings ∧ node.parent is defined then
10:    for sibling in node.parent.children do
11:      contextNodes.add(sibling)
12:   return contextNodes
```

Algorithm 4 Find descendants for a node until given depth

```
1: function FINDDESCENDANTS(node, maxDepth, depth)
2:   if depth ≥ maxDepth then
3:     return {}
4:   result ← node.children
5:   for child in node.children do
6:     descendants ← FindDescendants(child, maxDepth, depth+1)
7:     result ← result ∪ descendants
8:   return children
```

クローン検出モデルの構築

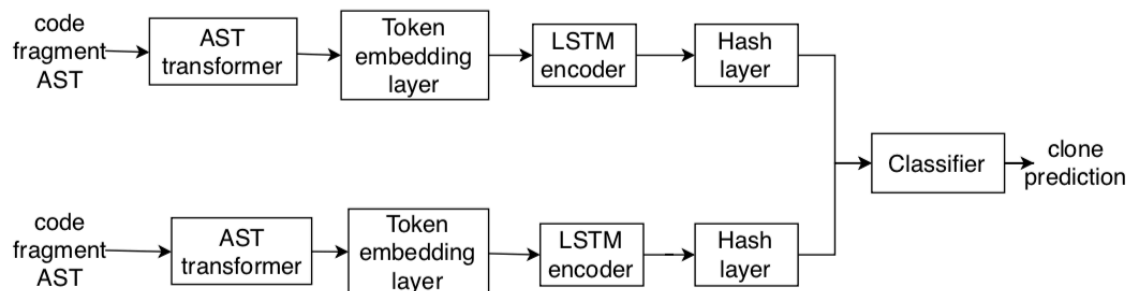


Fig. 4. Clone detection model overview

- Token embedding layerが訓練用データ生成に当たる
- LSTMを用いてASTの行列をベクトル化
- Hash layerで次元を減らす
- Classifierでシグモイド出力層を持つフィードフォワードニューラルネットワークを使用し、0から1までの類似度の値を出力

提案手法評価のためのRQ

- **RQ1**：JavaとPythonで書かれたプログラムの間で類似度を学習することはできるか
- **RQ2**：提案手法によって生成されたベクトル表現は類似度の学習能力を向上させることができるか

データセット：訓練データ生成用

TABLE I
TOKEN-LEVEL VECTORS GENERATION DATASET METRICS

	Java	Python
Projects count	1,027	879
Files count	476,685	131,506
Lines count	80,367,840	55,796,594
Tokens count	301,930,231	89,757,436

- skip-gramモデル用訓練データを生成するためのデータセット
- 以下の条件を満たすプロジェクトをGitHubから収集
 - フォークされていない
 - 100KBから100MBまでのサイズ
 - 非ウィルスライセンス（BSDやMITなど）を用いている

データセット：訓練用コードクローン

- プログラミングコンテストサイトから収集
 - 1つの問題に対して異なる言語で解決案が出される
 - それらの解決案は複数のユーザによって出されるため、機能は同じであるが書き方が異なるコードが集まる
- 下記のサイトで初学者向けにコンテストを開催

TABLE II
CLONE DETECTION DATASET METRICS

	Java	Python
Number of problems	576	
Avg. solutions / problem	36	41
Files count	20,828	23,792
Avg. lines / file	46	13
Avg. tokens / file	324	76

ベクトル生成

- 右図はベクトル表現を平面状にプロットしたもの
- k平均法を用いた結果宣言, 式, 文は正しくクラスタリングされていることがわかる
- 意味もある程度は保持できていることがわかる

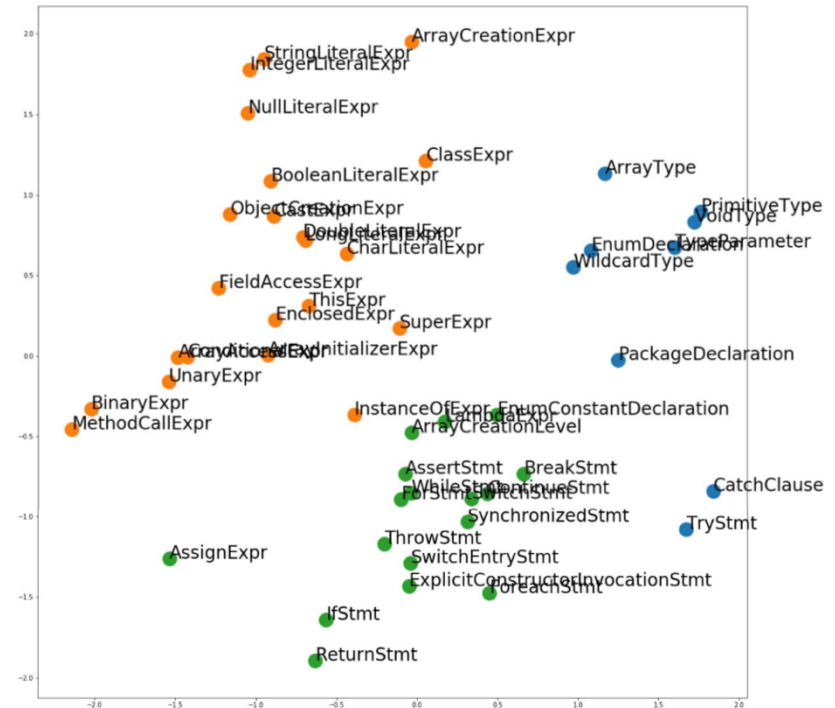


Fig. 5. Java token-level vectors projection in 2D

モデルの学習とテスト

- **80%**のデータは訓練用, **10%**のデータは交差検証用, 残りの**10%**は最終的なテストデータ
- **5つ**のコード片のうち**1つ**だけコードクローンにし, 残りはネガティブサンプルとする

モデルの学習とテスト

- 提案したモデルの性能を評価するために以下のモデルを設定
 - BaseLine：字句を順番通り skip-gram に学習させる
 - Pre-trained token vectors：提案手法
 - Randomly initialized token vectors：学習済みベクトル表現をランダムに初期化したベクトルに置き換える
 - Pre-trained token vectors, no values：変数の型を区別しない

モデルの学習とテスト

- BaseLineよりPre-trainedのF値が複数言語で13%, 単体言語で12%向上
 - RQ1に積極的な回答ができる
- Randomly initialized token vectorsよりPra-trainedのF値が複数言語で5%, 単体言語で3%向上
 - RQ2に積極的な回答ができる

TABLE V
JAVA/PYTHON CLONE CLASSIFICATION RESULTS

Model	F1-score	Precision	Recall
Baseline	0.53	0.41	0.74
Pre-trained token vectors, no values	0.51	0.40	0.71
Randomly initialized token vectors	0.61	0.49	0.82
Pre-trained token vectors	0.66	0.55	0.83

TABLE VI
JAVA/JAVA CLONE CLASSIFICATION RESULTS

Model	F1-score	Precision	Recall
Baseline	0.65	0.50	0.92
Pre-trained token vectors, no values	0.69	0.56	0.90
Randomly initialized token vectors	0.74	0.65	0.85
Pre-trained token vectors	0.77	0.67	0.92

モデルの学習とテスト

TABLE VII
CROSS-LANGUAGE CLONE DETECTION RESULTS

Metric	Result
F1-score	0.32
Precision	0.19
Recall	0.90

- 与えられたコード片のすべての組み合わせの間の類似性を評価
- 再現率が高いが適合率が小さい
- 訓練段階でクローン1つにつきネガティブサンプルが4つしかなかったためFalse Positiveが増加

まとめ・所感

- まとめ

- 異なる言語間のコードクローン検出手法を構築
- 評価実験の結果、類似性を学習できるが実用的でないことがわかった

- 所感

- コードクローンの正解集合の収集方法が良いアイデアだと思った
- 言語ごとに正解集合を用意する必要があるので、かなり手間がかかると思った