



Detecting Code Clones with Graph Neural Network and Flow-Augmented Abstract Syntax Tree

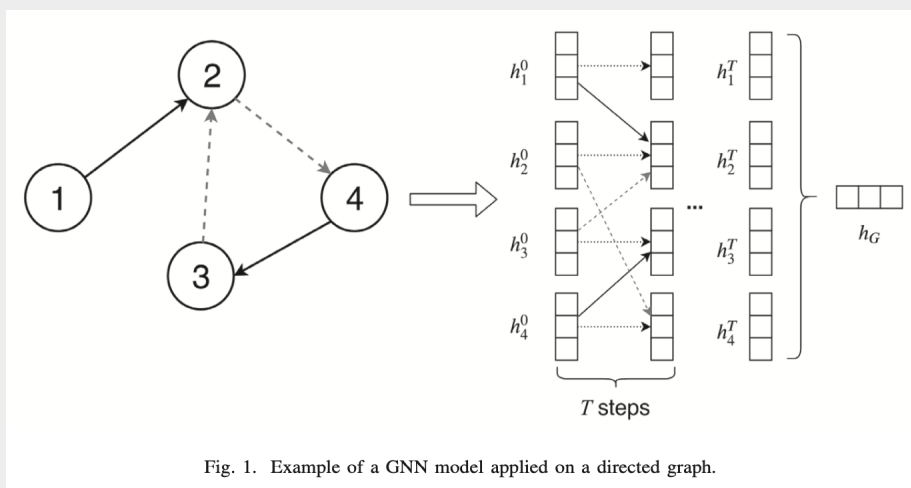
(グラフニューラルネットワークとフロー拡張抽象構文木を用いる
コードクローンの検出)

Wenhan Wang, Ge Li, Bo Ma, Xin Xia, Zhi Jin

2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)

グラフニューラルネットワーク (Graph Neural Network)

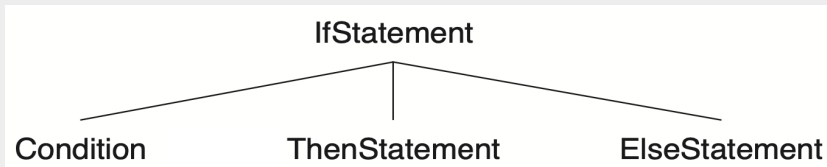
- GNNはグラフで表現されたデータを適用し, ディープラーニングの一つの手法
- GNNの目的は, 各ノードの状態埋め込みを学習することであり, グラフ全体の埋め込みを学習することもある



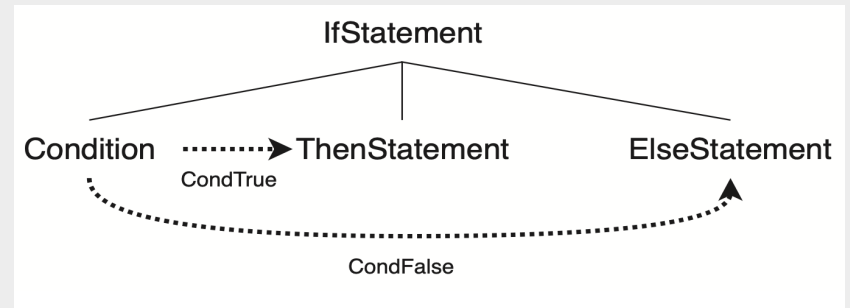
フロー拡張抽象構文木

(Flow-augmented Abstract Syntax Tree, FA-AST)

- 抽象構文木(Abstract Syntax Tree, AST)に制御フローとデータフローを追加すること



a) If文の抽象構文木



b) 制御フローが追加された
抽象構文木

背景

- コードクローンには4つのタイプがある



タイプ1



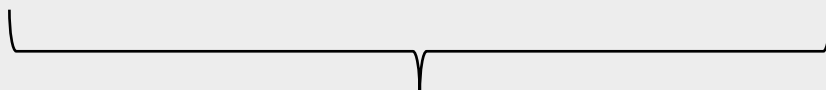
タイプ2



タイプ3



タイプ4



構文の類似性

意味の類似性

構文的に非常に異なるクローンを含む

ほとんどのクローン検出アプローチで検出するのが最も難しい

タイプ1 (T1) : 空白とコメントの違いを除き、構文的に同じコードフラグメント

タイプ2 (T2) : タイプ1のクローンの違いに加え、識別子名とリテラル値の違いを除き、構文的に同一のコードフラグメント

タイプ3 (T3) : タイプ1とタイプ2のクローンの違いに加え、ステートメントレベルで異なるが、構文的に類似したコードフラグメント (ステートメントが追加/変更/削除されている)

タイプ4 (T4) : 構文的に異なるが、同じ機能を持っているコードフラグメント

背景

- 意味の類似点を捉えようとして,いくつかの新しいディープラーニングベースのアプローチがある
- これらのアプローチのほとんどには,2つのステップが含まれる
 - ニューラルネットワークを用い,コードフラグメントのベクトル表現を計算する
 - 2つのコードベクトル表現の類似性を計算し,クローンを検出する
- これらのアプローチは通常,抽象構文木 (Abstract Syntax Tree, AST) を入力として使用される
- ただし,ASTには,制御フローやデータフローなどの意味的情報が含まれていない

目的

- ASTからの構文的情報と意味的情報がグラフ表現形式で反映できるプログラムを構築すること
- 作成したグラフからコードクローンを検出するために, グラフニューラルネットワーク (GNN) を使用してコードクローンを検出する新しいアプローチを提案する

アプローチ

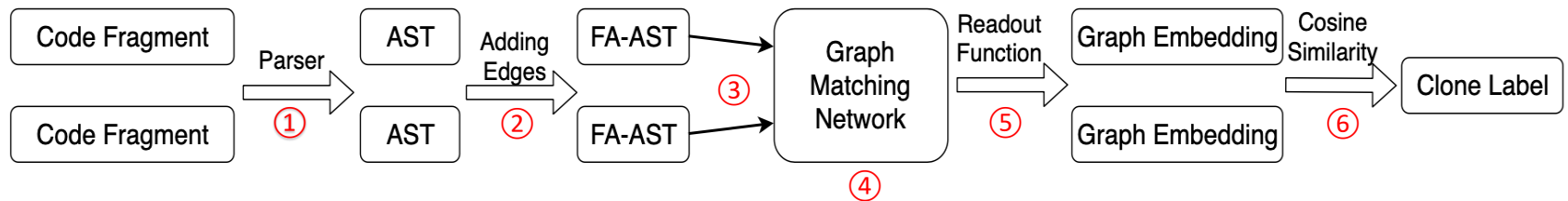


Fig. 2. The overview of our approach.

1. ASTに解析する
2. ASTに制御とデータフローを表すエッジを追加し, FA-ASTを作成
3. FA-ASTノードとエッジの埋め込みを初期化する
4. グラフマッチングネットワークは, すべてのノードのベクトル表現を計算
5. 各FA-ASTは, 読み出し関数を用いて全てのノードベクトルを組み合わせ, グラフレベルのベクトルで表現する
6. ベクトル表現を取得した後, 2つのベクトルのコサイン類似度を使用し, 類似度を計算

アプローチ

- 2種類のグラフニューラルネットワークを使用する:

- グラフ埋め込みモデル:

- Gated Graph Neural Network (GGNN) を使用して, グラフの埋め込みを学習

- Graph Matching Network(GMN):

- 一対のグラフの埋め込みを共同で学習できる
- 従来のGNNプロセスと比較し, GMNはさらに2つのグラフからノード間のクロスグラフアテンション (cross-graph attention) を計算する

アプローチ

- 図7は、GGNNとGMNの違いを示している

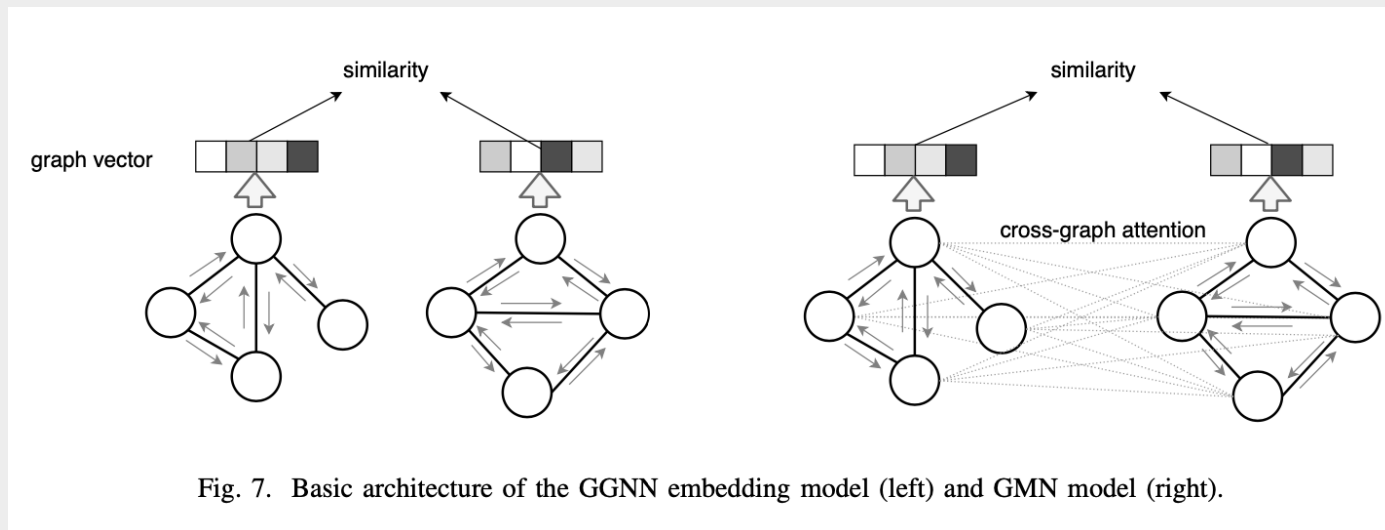


Fig. 7. Basic architecture of the GGNN embedding model (left) and GMN model (right).

実験

- 本論文のアプローチは2つのデータセットで評価された：
 - Google Code Jam (GCJ) と BigCloneBench
- GCJデータセットは, 12の異なるコンテスト問題からの1,669 Java ファイルで構成されている
- BigCloneBenchデータセットは, 9,134のJava ファイルで構成されている
- データセットは, 訓練、検証, テストセット用に8:1:1に分割された

TABLE II
BASIC INFORMATION OF TWO DATASETS

	GCJ	BigCloneBench
Code fragments	1,669	9,134
Average lines of code	58.79	32.89
Average number of nodes	396.98	241.46
Vocabulary size	8,033	77,535
True clone pairs	275,570	336,498
False clone pairs	1,116,376	2,080,088

結果

- Google Code Jamの結果：
 - 表4は, GCJの結果を示している
 - 提案されたアプローチは他のアプローチよりも優れている

TABLE IV
RESULTS ON THE GCJ DATASET

Model	Precision	Recall	F1
Deckard	0.45	0.44	0.44
RtvNN	0.20	0.90	0.33
ASTNN	0.98	0.93	0.95
FA-AST+GGNN	0.96	1.0	0.97
FA-AST+GMN	0.99	0.97	0.98

結果

- BigCloneBenchでの結果：
 - 表5はBigCloneBenchの結果を表示している
 - 本論文のアプローチは, ほとんどのアプローチより高い再現率とF値を持っている

TABLE V
RESULTS ON THE BIGCLONEBENCH DATASET

Model	Precision	Recall	F1
Deckard	0.93	0.02	0.03
RtvNN	0.95	0.01	0.01
CDLH	0.92	0.74	0.82
ASTNN	0.92	0.94	0.93
FA-AST+GGNN	0.85	0.90	0.88
FA-AST+GMN	0.96	0.94	0.95

結果

- 異なるクローンタイプを個別に検出できるかを調べた
- 表6は, BigCloneBenchの全てのクローンタイプの結果を表示している
- 本研究のアプローチ (FA-AST + GMN) をASTNNと比較する

TABLE VI
RESULTS ON DIFFERENT CLONE TYPES IN BIGCLONEBENCH

Type	ASTNN			FA-AST+GMN		
	Precision	Recall	F1	Precision	Recall	F1
T1	100	100	100	100	100	100
T2	100	100	100	100	100	100
ST3	100	99.6	99.8	100	99.6	99.8
MT3	100	97.9	98.9	100	96.5	98.2
WT3/T4	93.3	92.2	92.8	95.7	93.5	94.6

コードクローン例

- 本研究のアプローチ (FA-AST + GMN) が正しく予測されたが, ASTNNはそうではなかった
- 図11 (a) と図11 (b) はBigCloneBenchの真のクローンペアに属し, 両方のコードフラグメントがファイルコピー機能を実装している
- 予測された類似性スコアは
 - ASTNN : 5.8e-07
 - FA-AST : 0.94

```
public static void copyFile(File source, File dest) throws IOException {  
    FileChannel in = null, out = null;  
    try {  
        in = new FileInputStream(source).getChannel();  
        out = new FileOutputStream(dest).getChannel();  
        in.transferTo(0, in.size(), out);  
    } catch (FileNotFoundException fnfe) {  
        Log.debug(fnfe);  
    } finally {  
        if (in != null) in.close();  
        if (out != null) out.close();  
    }  
}
```

(a)

```
private void createButtonCopyToClipboard() {  
    buttonCopyToClipboard = new Button(shell, SWT.PUSH);  
    buttonCopyToClipboard.setText("Co&py to Clipboard");  
    buttonCopyToClipboard.setLayoutData(SharedStyle.relativeToBottomRight(buttonClose));  
    buttonCopyToClipboard.addSelectionListener(new SelectionAdapter() {  
        @Override  
        public void widgetSelected(final SelectionEvent event) {  
            IOUtils.copyToClipboard(Version.getEnvironmentReport());  
        }  
    });  
}
```

(b)

Fig. 11. Example of a true clone pair in BigCloneBench which FA-AST+GMN correctly predicted as true while ASTNN wrongly predicted.

まとめ

- コードクローンの検出については多くの研究があるけど, 意味的に似ているクローンを検出できるのはごくわずか
- コードクローン検出のために, 制御フローとデータフロー情報を活用する新しいアプローチを提案する
- このアプローチは, 2つの異なるGNN (GGNNとGMN) をフロー拡張AST (FA-AST) に適用する

所感

- ディープラーニングは難しいと思うけど, この論文を読んでいる時いろいろ勉強になった