競技プログラミングのデータを用いたクローン検出

ペレズ ダニエル, 千葉 滋
東京大学

クローン検出について

クローン検出とは

複数のプログラムで重複したコードを見つけ出すタスク

クローン検出の主な手法

・トークンベースの手法: トークンの比較
・ASTベースの手法: 構文木の比較

複数の言語間の場合

入力のプログラミング言語が違う場合は
・共通のトークンが少ない
・ASTの構造が違う
→ 既存手法をそのまま使うことが困難

複数の言語間でのクローンの例

以下のコードは同じ機能を実装しているが、クローン検出が困難

Javaのgroup byメソッド

```java
public Map<String, List<Record>>
    groupRecords(List<Record> records) {
  Map<String, List<Record>> grouped =
    new HashMap<>();
  for (Record record: records) {
    if (!grouped.containsKey(
          record.getState())) {
      grouped.put(record.getState(),
                  new ArrayList<Record>());
    }
    grouped.get(record.getState()).add(record);
  }
  return grouped;
}
```

Pythonのgroup by関数

```python
def group_records(records):
    result = {}
    for record in records:
        bucket = result.setdefault(
            record.state, [])
        bucket.append(record)
    return result
```

教師あり学習を用いた手法の提案

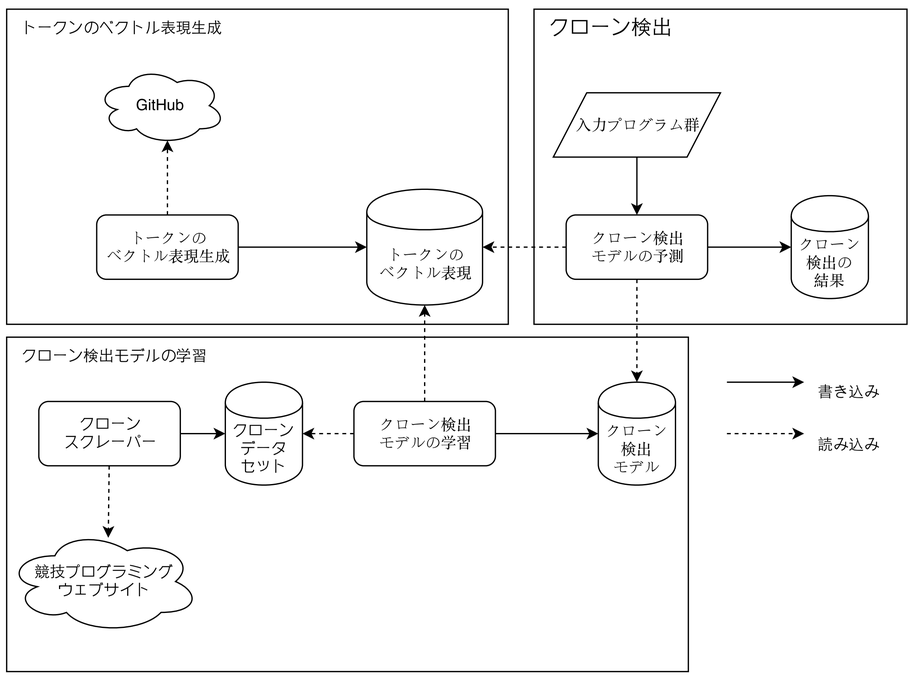
キーアイデア

・ASTの構造のベクトル表現を学習
  – 学習データとして競技プログラミングのデータを利用
・ASTのベクトル表現の比較でクローン検出

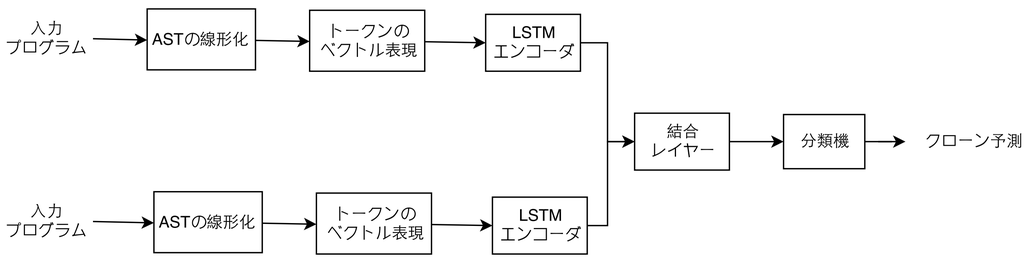
流れ

1. トークンのベクトル表現の学習
2. クローン検出モデルの学習
3. 学習済みのモデルでクローン検出

クローン検出システム



モデルの構造



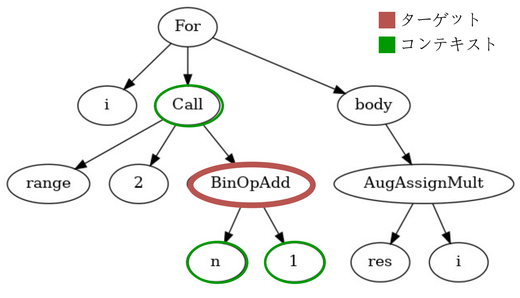
Tree-based skipgramアルゴリズム

ASTの木構造の情報を使うようにSkipgramモデルを拡張

アルゴリズムの流れ

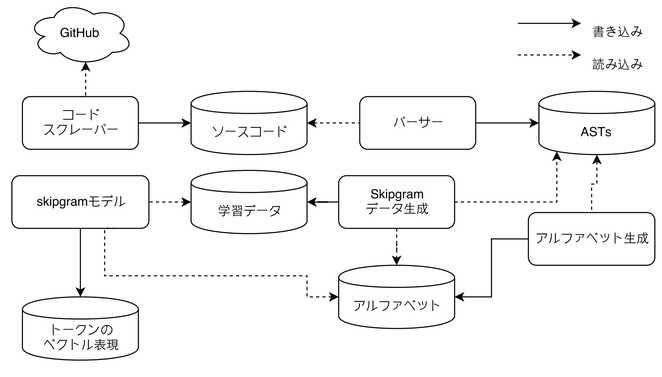
1. アルファベットの作成
2. ASTからターゲットとコンテキストの組の集合を作成
3. 順伝播型ニューラルネットワークで学習

Tree-based skipgramの入力例

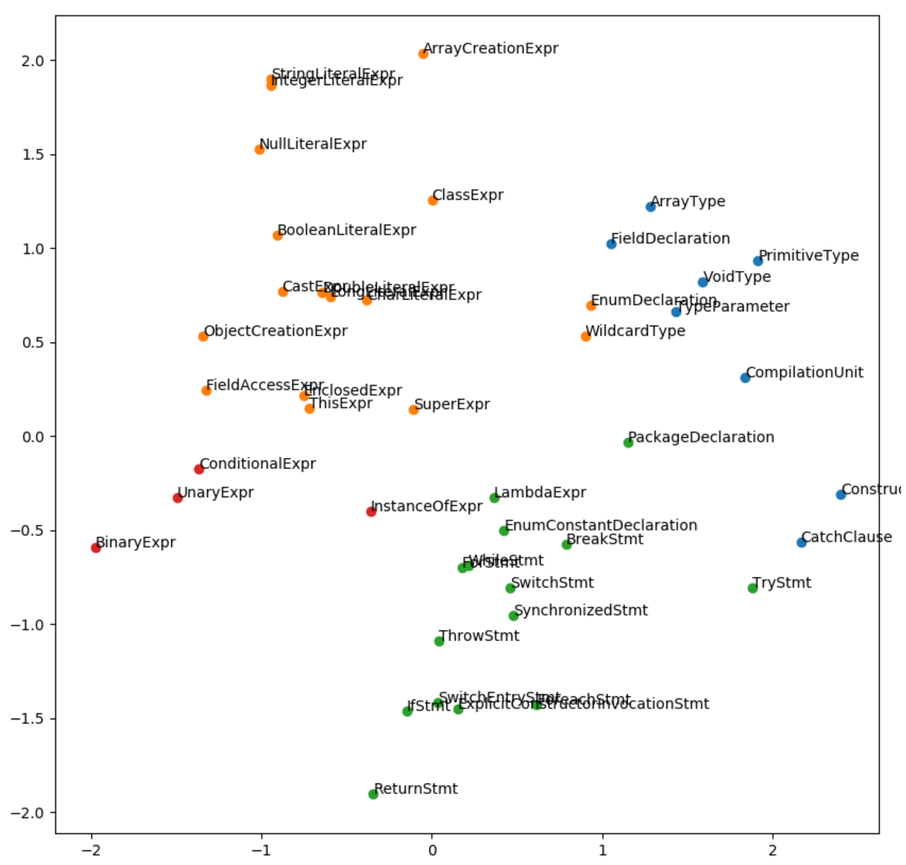


Pythonの階乗関数のメインループ

トークンのベクトル表現の学習機



Javaのトークンのベクトル表現



クローン検出の実験結果

Java-Javaのクローン検出よりJava-Pythonのクローン検出の方が難しい

トークンの	Python			Java		
ベクトル表現	F1値	Precision	Recall	F1値	Precision	Recall
学習済み	0.66	0.55	0.83	0.77	0.67	0.92
未学習	0.61	0.49	0.82	0.74	0.65	0.85
変数名なし	0.51	0.40	0.71	0.69	0.56	0.90

クローンの割合を20%にして学習とテストした時の結果

今後の課題

・AST構造により適切なモデルを利用: LSTM → Gated Graph Sequence NN
・システムの実行時間を線形時間へ: $\mathcal{O}\left(n^2\right) \rightarrow \mathcal{O}\left(n\right)$
  – ハッシュレイヤーの追加
  – インデックスの追加