THE IEICE TRANSACTIONS ON INFORMATION AND SYSTEMS (JAPANESE EDITION)

電子情報通信学会 論文誌 D 情報・システム

VOL. J104-D NO. 9
SEPTEMBER 2021

情報・システムソサイエティ

一般社団法人 電子情報通信学会
THE INFORMATION AND SYSTEMS SOCIETY
THE INSTITUTE OF ELECTRONICS, INFORMATION AND COMMUNICATION ENGINEERS

論　文

回帰モデルを用いたコードクローン検出手法の提案と汎化性能の評価

藤原　裕士[†,††a)]　　森　　彰[††]　　井上　克郎[†,††]

An Approach to Detecting Semantic Code Clone using Regression Model and Evaluating Versatility

Yuji FUJIWARA[†,††a)], Akira MORI[††], and Katsuro INOUE[†,††]

あらまし　意味的コードクローン検出の技術はソフトウェア開発や保守において重要である．近年，深層学習を用いた意味的コードクローン検出手法が多く提案され，高い検出精度を記録しているが，コードクローンの学習方法についての問題や，深層学習モデルの汎化性能に関する評価の問題がある．そこで本研究では，回帰モデルを用いたコードクローン検出手法を提案し，その汎化性能について評価する．提案手法では，深層学習を用いる既存の意味的コードクローン検出手法で利用されている 2 値分類モデルの代わりに回帰モデルを利用し，教師あり学習の目的変数に抽象構文木の類似度を設定することで，コードクローンの学習方法を変更する．評価実験では，学習データセットと評価データセットをそれぞれ異なるプロジェクトから作成し，四つの深層学習モデルにおける 2 値分類モデルと回帰モデルの F 値と AUC を評価した．その結果，四つ中三つの深層学習モデルにおいて，回帰モデルに変更することで F 値と AUC が向上した．

キーワード　コードクローン，深層学習，回帰モデル，汎化性能

1. まえがき

近年，大規模なソフトウェアを効率的に開発することが求められる中，意味的コードクローン検出技術は，ソフトウェアの開発や保守において重要な役割を果たす．意味的コードクローン検出技術を用いて，過去に開発されたソフトウェアの中から開発者が求める機能をもったソースコードを効率的に検索し再利用することで，既存ソフトウェアを効率的に利用したソフトウェア開発を行うことができる．また，ソフトウェアの保守性を低下させる原因となるコードクローンを検出し集約することで，ソフトウェアの保守性を向上させることができる．

現在まで，深層学習を用いた意味的コードクローン検出手法が提案されている [1]〜[6]．例えば Zhang ら [1] は，独自に開発した深層学習モデルである ASTNN を用いて意味的コードクローン検出に取り組んでいる．評価実験では，オープンジャッジのソースコードを用いたデータセットと大規模コードクローンベンチマーク BigCloneBench [7] に対して ASTNN を用いたコードクローン検出を実行し，オープンジャッジデータセットに対しては 0.955，BigCloneBench に対しては 0.938 の F 値を記録している．

このように近年，深層学習を用いた意味的コードクローン検出手法が多く提案されている．しかし，既存手法には二つの問題点がある．一つ目は，コードクローンの学習方法である．コードクローンの類似度は様々であり，構文的に類似していて簡単に識別できるコードクローンがある一方，構文的な類似度が低く，人によってコードクローンかどうか意見が別れるコードクローンも存在する．しかし，多くの既存手法では 2 値分類モデルが用いられているため，与えられたコード片のペアは，コードクローンか非コードクローンかの 2 択で表現される．このように，明らかなコードクローンも人によって意見が別れるコードクローンも全て同じ重みで表現されるため，コードクローンの類似度情報が欠落した状態で学習を行っている．そのため，コードクローン検出精度が低下している可能性がある．二つ目は，深層学習モデルの汎化性能が正

† 大阪大学，吹田市
Osaka University, Suita-shi, 565–0871 Japan
†† 産業総合技術研究所，池田市
Industrial Science and Technology, Ikeda-shi, 563–8577 Japan
a) E-mail: y-fujiwr@ist.osaka-u.ac.jp
DOI:10.14923/transinfj.2020JDP7073

しく評価できていない点である．既存研究の評価実験では，同一プロジェクトから学習データセットと評価データセットを構築していることが多い．この場合，二つのデータセットに依存関係があるため，深層学習モデルが学習プロジェクトに特化している可能性を否定できず，深層学習モデルの汎化性能についての評価は不十分である．

そこで本研究では，回帰モデルを用いたコードクローン検出手法を提案し，その汎化性能について評価する．この手法では，コードクローンの学習方法を改善するために，2 値分類モデルではなく回帰モデルを利用し，教師あり学習に使用する目的変数をコード片の類似度に変更する．このように学習させるコードクローンの重みを類似度に基づいて変更することで，コードクローンの類似度情報を利用した学習を行う．

評価実験では，未学習データに対する深層学習モデルの挙動を確認し汎化性能を評価するために，学習データセットと評価データセットを同一プロジェクトから作成するのではなく，依存関係のない二つのプロジェクトから作成した．そして，ASTNN などの四つの手法で用いられている深層学習モデルを 2 値分類モデルから回帰モデルに変更し，汎化性能を比較した．その結果，四つの内三つの深層学習モデルの汎化性能が向上することが分かった．例えば ASTNN の場合，学習データセットとして BigCloneBench のコードクローンを利用し，評価データセットとしてオープンジャッジのソースコードを利用することによってデータセット間の依存関係を取り除くと，F 値は 0.938 から 0.392 に低下した．しかし，2 値分類モデルから回帰モデルに変更することで F 値は 0.392 から 0.694 に向上した．

本論文の貢献は以下の 2 点である．

- 2 値分類モデルから回帰モデルに変更し教師あり学習の目的変数をコード片の類似度にすることで，学習におけるコードクローンごとの重みを設定した．その結果，四つの深層学習モデルのうち三つで汎化性能が向上することを確認した．
- 未学習データに対する深層学習モデルの挙動を確認し，汎化性能を評価するために，学習データセットと評価データセットを同一プロジェクトから作成するのではなく，依存関係のない二つのプロジェクトから作成した．

以降，**2.** では本論文の背景として，コードクローン，コードクローン検出に用いられる代表的な深層学習モデルである Siamese モデル，深層学習モデルの汎化性能の評価を阻害する原因であるデータセットの依存関係について説明する．**3.** では，コードクローンの類似度を学習に反映するための提案手法について説明する．**4.** では，深層学習モデルの汎化性能をより正確に評価するための実験について説明する．**5.** では，深層学習を用いた意味的コードクローン検出の関連研究を紹介する．最後に，**6.** で本研究のまとめについて述べる．

2. 背　　景

2.1 コードクローン

コードクローンとは，互いに一致または類似したコード片のことである．一般的に，コードクローンの存在は，ソフトウェアの保守を困難にする要因であるといわれている [8]．コードクローンの主な発生要因として，既存ソースコードのコピーアンドペーストによる再利用，定型処理（イディオム）による発生，ツールによるソースコード自動生成，偶然の一致による発生などが挙げられる [9]．Roy らはコード片の違いの程度に基づき，コードクローンを以下の 4 タイプに分類している [10]．

タイプ 1 空白やタブの有無，コーディングスタイル，コメントの有無などの違いを除き完全に一致するコードクローン

タイプ 2 タイプ 1 の違いに加え，識別子名，変数の型などが異なるコードクローン

タイプ 3 タイプ 2 の違いに加え，文の挿入や削除，変更などが行われているコードクローン

タイプ 4 類似した処理を実行するが，構文上の実装が異なるコードクローン

一般的に “意味的コードクローン” は，構文的な類似度が低いコードクローンを指す．また，Roy らは既存研究 [7] において，構文的な類似度が 50% 未満のタイプ 3 クローンとタイプ 4 クローンをまとめて扱っている．そのため本論文では，“意味的コードクローン” を，構文的な類似度が 50% 未満のタイプ 3 クローン及びタイプ 4 のコードクローンと定義する．

また，Roy らは，コードクローン検出ツールを評価するための大規模なデータセットとして，BigCloneBench（以降，BCB）[7] を作成した．このデータセットは，様々なオープンソースソフトウェアに存在するコードクローンを収集したものであり，タイプ 1 からタイプ

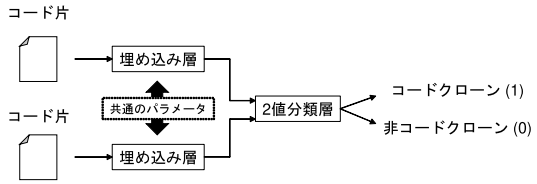

図 1　コードクローン検出への Siamese モデル適用例

4 まで，様々な類似度のコードクローンが含まれている．BCB は現在に至るまで様々なコードクローン検出ツールの評価に利用されている．また，深層学習を用いてコードクローン検出を行う研究では，BCB は深層学習モデルの学習データセットとして利用されることもある．

また，近年では，ソースコード記述以外の情報を利用して意味的コードクローンデータセットを構築することがある．Roy ら [11] は大規模 Q&A サイト StackOverflow(注1) に注目した．同じ質問に寄せられた正しく動作する回答ソースコードは互いに意味的コードクローンであるという仮定の下，意味的コードクローンベンチマーク SemanticCloneBenchmark を提案した．Zhao と Huang [2] は，オープンジャッジシステムにおいて，ある設問に対して提出されたコード片はその設問を解決するための機能をもつことに注目した．そして，同じ設問に提出されたコード片のペアは意味的コードクローンであり，互いに異なる設問に提出されたコード片のペアを非コードクローンであるという仮定の下でデータセットを作成し，Deepsim の評価実験に利用した．Kamp ら [12] は，コード片を説明するコメントである JavaDoc の文章類似度が高いコード片のペアは意味的コードクローンであるという仮定の下，意味的コードクローンデータセット SeSaMe を提案した．

2.2　Siamese モデル

近年，深層学習を用いたコードクローン検出手法が多く提案されている [1]〜[6]．コードクローン検出に深層学習を適用する方法の一つに Siamese モデル [13] がある．このモデルは，二つの入力の類似度を推測するための教師あり学習モデルである．

コードクローン検出に対する Siamese モデルの適用例を図 1 に示す．Siamese モデルはニューラルネットワークの一つであり，共通のパラメータをもつ二つの埋め込み層と，一つの 2 値分類層から構成される．埋め込み層は，コード片が入力されるとコード片の特徴ベクトルを出力する層である．二つの埋め込み層は共通のパラメータをもつため，同じコード片が入力された場合に同じベクトルを生成することが，Siamese モデルの特徴である．2 値分類層は，二つの特徴ベクトルが入力されると，二つの特徴ベクトルに対応するコード片がコードクローンであれば 1，非コードクローンであれば 0 を出力する層である．

2.3　データセット間の依存関係

データセット間に依存関係がある状態とは，同じデータセットを分割するなどの方法で学習データセットと評価データセットを構築することにより，データの特徴がデータセット間で酷似している状態のことである．学習・評価データセット間に依存関係があると，深層学習モデルが過学習を起こしていた場合でも，評価データセットに対する予測精度が高くなる．そのため，深層学習モデルの汎化性能を正しく評価することができない．ソースコードの特徴として，識別子名を開発者が自由に定義できるため，ソースコードを学習する段階で全ての識別子名を網羅することは難しいという点がある．そのため，未学習の識別子名が評価データセットに含まれている場合でも正しい予測を行うことができるかどうかという点は，深層学習モデルの汎化性能を評価する上で非常に重要である．しかし，同じデータセットを分割して学習・評価データセットを構築した場合，評価データセット内に出現する未学習識別子名が少ない可能性があるため，深層学習モデルの汎化性能を正しく評価できていない．

例えば ASTNN [1] の場合，BCB に含まれるコードクローンと非コードクローンをそれぞれ 3:1:1 の割合に分割し，学習データセット・検証データセット・評価データセットとすることで，適合率・再現率・F 値はそれぞれ 0.998，0.884，0.938 を記録している．しかし，筆者らが BCB のコードクローンを学習した ASTNN モデルを再現し，Deepsim [2] の評価実験に用いられたオープンジャッジデータセットを用いて評価したところ，適合率・再現率・F 値はそれぞれ 0.521，0.314，0.392 を記録した．このように，学習データセットとの間に依存関係がある評価データセットを用いた場合には検出精度が高くても，依存関係のない評価データセットを用いると検出精度が大きく低下する場合がある．

(注1)：https://stackoverflow.com/

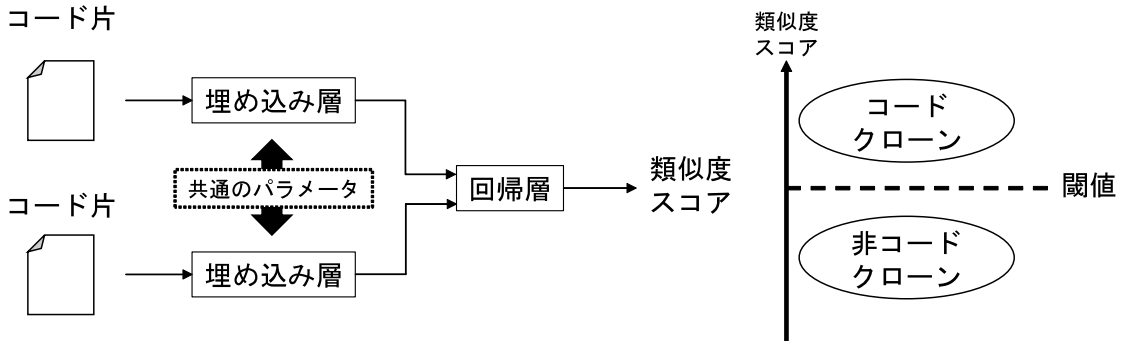

図 2　提案モデル

3. 提 案 手 法

本章では，回帰モデルを用いたコードクローン検出手法を提案する．多くの既存の意味的コードクローン検出手法では，入力された二つのコード片がコードクローンか判定するために 2 値分類モデルを用いている．それに対して提案手法では，2 値分類モデルではなく回帰モデルを利用し，目的変数を抽象構文木（Abstract Syntax Tree，以降，AST）の類似度に変更することで，コードクローンの類似度を学習に反映する．このようにすることで，学習させるコードクローンの重みに類似度を反映することができる．

3.1　回帰モデルの適用

多くの既存の意味的コードクローン検出手法では 2 値分類モデルが用いられている．このモデルは，入力されたコード片のペアがコードクローンであれば 1 を，非コードクローンであれば 0 を出力する．2 値分類モデルでは全てのコードクローンは同じ重みで表現される．しかし，コードクローンには類似度の違いがある．多くの人がコードクローンだと判断できるコードクローンがある一方，人によってコードクローンかどうか意見が分かれるコードクローンもある．それらのコードクローンを全て同じ重みで扱っているため，コードクローンの類似度情報が欠落し，深層学習モデルのコードクローン検出性能を低下させている可能性が考えられる．

そのため，本研究では 2 値分類モデルの代わりに回帰モデルを用いる．2 値分類モデルの目的変数は整数であるのに対し，回帰モデルの目的変数は実数である．そのため，コードクローンの類似度を目的変数に設定し，コードクローンの類似度情報を欠落させることなく学習に利用することができる．

一般的な意味的コードクローン検出モデルを 2 値分類モデルから回帰モデルに変更した例を図 2 に示す．図 2 の提案モデルでは，2 値分類層の代わりに回帰層を用いる．具体的には，学習に使用する損失関数を the Binary Cross Entropy から the Mean Squared Error に変更する．そのため，深層学習モデルの出力は整数ではなく実数となる．そして，入力されたコード片のペアがコードクローンかどうかを，深層学習モデルが出力する類似度スコアに基づいて判定するために，しきい値 T を設定する．深層学習モデルが出力した類似度スコアがしきい値 T より大きい場合はコードクローン，小さい場合は非コードクローンであると判定する．

このように，提案手法は損失関数の変更を行うという単純な手法であるため，既存の深層学習モデルに対して容易に適用できる手法である．

3.2　目的変数の変更

回帰モデルを用いて教師あり学習を行う場合，目的変数には実数を設定する必要がある．提案手法では，コードクローンの場合は以下で説明する二つのコード片の AST 類似度を目的変数に設定し，非コードクローンの場合は 0.0 を目的変数に設定する．この点が，単に AST 類似度が高いペアをコードクローンだと判定する手法との違いである．提案手法の構成をとることによって，“意味的コードクローンではない” と学習データ内でラベリングされているペアは，たとえ AST 類似度の値が大きくても，0 が出力されるように深層学習モデルの学習が進行する．これにより，意味的コードクローンに対するモデルの出力スコアは，非コードクローンに対する出力スコアよりも高くなるため，意味的コードクローンを検出することができる．

本研究では AST 類似度を算出するツールを，Zhang と Shasha が提案したアルゴリズム [14] を元に実装し

た．Zhang と Shasha のアルゴリズムは，順不同の木のペア (A, B) が与えられたとき，最小編集コストで木 A を木 B に変換する編集シーケンスを作成するためのものである．編集シーケンスは，ノードのラベル変更，木へのノード挿入，ノード削除の三つのノード編集操作の列である．また，編集コストは，各ノード編集操作のコストの合計で定義され，通常，各操作のコストは 1 である．Zhang と Shasha のアルゴリズムは最小編集コストでの編集シーケンスの検出は保証されているが，時間計算量はノード数の 2 乗に比例するため，大きな木に対しては時間効率が悪くなる．

そこで本研究では，Hashimoto と Mori によって提案された編集コスト近似手法 [15] を用いて効率的に木の編集シーケンスを計算できるようにした．また，ノードの移動を新たに編集操作として認めることで，ソースコードの AST に特化した編集シーケンスを作成するツールを実装した．そして，このツールで得られた編集シーケンスを用いて AST 類似度を定義した．具体的には，ある AST のペア (A, B) の間の編集シーケンスを s，A, B のノード数を x, y，編集シーケンス s によって新たに挿入及び削除がされておらず，何も編集操作が適用されていないか，ラベル変更やノード移動の操作のみが適用されているノードの数を z としたとき，ペア (A, B) の類似度 S を次のように定義した．

$$S(A, B) = \frac{2z}{x + y}$$

また，この AST 類似度 S を使用して，各コード片のペアに対する目的変数の値 V を次のように定義した．ここで，コード片 a, b の AST は A, B である．

$$V(a, b) = \begin{cases} S(A, B) & (a, b \text{ がコードクローン }) \\ 0 & (a, b \text{ が非コードクローン }) \end{cases}$$

そして，コード片の組 (a, b) を説明変数，$V(a, b)$ を目的変数とした教師あり学習を行うことで，意味的コードクローン検出モデルを作成する．

4. 評価実験

深層学習を用いた意味的コードクローン検出において，2 値分類モデルから回帰モデルに変更することがコードクローン検出精度や深層学習モデルの汎化性能にどのような影響を与えるか確認するために，本研究では評価実験を行った．ここで，実験対象モデルの一つである ASTNN [1] のコードクローン検出粒度がメソッド単位であるため，本評価実験のコードクローンの粒度はメソッド単位とする．本評価実験では，深層学習モデルの汎化性能を正確に評価するため，**2.3** で説明した依存関係をもたないように学習データセットと評価データセットを構築した．また，本評価実験を行った環境を表 1 に示す．

4.1 評価尺度

4.1.1 適合率・再現率・F 値

まず適合率とは，コードクローン検出ツールがコードクローンであると判定したコード片ペアのうち，評価データセット中に存在するコードクローンの割合である．次に再現率とは，評価データセット中に存在するコードクローンのうち，コードクローン検出ツールがコードクローンだと判定することができた割合である．最後に F 値とは，適合率と再現率の調和平均である．これら三つの評価尺度は一般的にコードクローン検出ツールの評価尺度として用いられているため，本評価実験でもこれら三つの評価尺度を用いる．

4.1.2 AUC

AUC (Area Under the Curve) [16] とは，受信者操作特性（Receiver Operating Characteristic, 以降，ROC）曲線の下の面積であり，分類モデルの評価尺度として用いられる．ここで ROC 曲線とは，“1-再現率” を横軸，“適合率” を縦軸に設定し，しきい値を連続的に変化させて “1-再現率” と “適合率” の交点をプロットしていき，プロットされた点を結んだ曲線である．ROC 曲線を引く際にしきい値を変化させる必要があるので，AUC は一見すると 2 値分類モデルに適用できないように思える．しかし，2 値分類モデルでもモデルの内部で類似度スコアは計算されているため，類似度スコアを抽出ししきい値を変化させることで AUC を適用することができる．AUC の値域は 0 から 1 の実数であり，1 に近いほど，適切なしきい値を設定した場合の F 値が高く，分類性能が高いことを表す．コードクローン分野において広く用いられる評価尺度である適合率・再現率・F 値に加えて，分類モデルの評価尺度として一般的な AUC を用いることによって，より多

表 1 実験環境

OS	macOS Catalina 10.15.7
CPU	8-Core Intel Xeon W 3.2GHz
メモリ	128GB DDR4-2666
深層学習ライブラリ	PyTorch(注2)

(注2)：https://www.pytorch.org/

角的にコードクローン検出モデルを評価することができる．

4.2 データセット

既存の意味的コードクローン検出に関する研究では，一つのプロジェクトから学習データセットと評価データセットを作成し，深層学習モデルの学習と評価を行っている．しかし，この方法はデータセット間に**2.3** で説明した依存関係があるため，深層学習モデルの汎化性能を正しく評価できていない．そのため，本研究では，学習データセットと評価データセットをそれぞれ別のプロジェクトから作成し学習と評価を行うことで，既存研究よりも正確に深層学習モデルの汎化性能を評価する．具体的には，**2.1** で説明した五つのデータセットの内，目視確認が行われ，コードクローンに対するラベリングの正確さが保証されており，学習に十分な数のコードクローンを含むデータセットである BCB を用いて学習データセットを作成し，BCB 以外の四つのデータセットを用いて評価データセットを作成することで依存関係を排除した．以下に各データセットの詳細を述べる．

4.2.1 学習データセット

本評価実験では，手作業で正確にラベリングされているコードクローンが最も多いデータセットが BCB であることから，BCB を学習データセットとして利用する．最初に，ASTNN に対して BCB のデータベースに登録されているコードクローンと非コードクローンを学習させた．しかし，コードクローンの数に比べて非コードクローンの数が大幅に少なかったため，モデルは非コードクローンの特徴を学習できず，ほぼ全てのペアをコードクローンと判定するモデルが作成された．そのため，次に 5000 個の非コードクローンを学習データセットに追加して学習を行った．この非コードクローンは BCB に存在する二つのメソッドを組み合わせて作成した．その結果，ASTNN は非コードクローンの特徴を学習することができ，モデルが大部分のペアをコードクローンと判定する現象は発生しなかった．また，非コードクローンを更に追加し，コードクローンと非コードクローンを同じ数にして学習を行った．その結果，モデルはコードクローン特徴をうまく学習することができず，大部分のペアを非コードクローンと判定するモデルが作成された．そのため，本評価実験では，BCB に含まれるコードクローン，非コードクローンと，新たに作成した 5000 個の非コードクローンを学習データセットして利用する．学習データセットの内訳を表 2 に示す．

表 2　データセットの内訳

	コードクローン	非コードクローン
BCB	77508	24633
GCJ	274048	274048
SCB	861	861
SeSaMe	114	114
CSN	300	300

4.2.2 評価データセット

本評価実験では，GoogleCodeJam データセット，SemanticCloneBench，SeSaMe，CodeSearchNet データセットの計四つの意味的コードクローンベンチマークを評価データセットとして利用した．各評価データセットはユニークな視点から意味的コードクローンを定義している．また，構文的な類似度が高いことは，これらの評価データセットにおいてはコードクローンであることの必要条件ではないことから，BCB のコードクローンとは性質が異なる．そのため，本評価実験の学習データセットと評価データセットの間に依存関係は存在しないと考えられる．各評価データセットの内訳を表 2 に示す．

また，短いコード片は一般的にコードクローン検出ツールの評価に利用しない．短さの基準は様々であり，コードの行数の観点からは 5 行 [2], [17] や 10 行 [18]，トークン数の観点からは 50 トークン [3], [17] などがよく用いられる．コード片の行数は書き方に依存するため，本研究では 50 トークンを短さのしきい値として採用し，50 トークン以下のメソッドは評価データセットから取り除いた．

GoogleCodeJam データセット（以降，GCJ）[2] は，Zhao と Huang によって，オープンジャッジシステムに提出された回答ソースコードを用いて構築されたデータセットである．このデータセットは “ある同じ設問に対して提出されたメソッドはその設問を解決するための機能をもつため，互いに意味的コードクローンである” という仮定の下構築されている．また，非コードクローンは “互いに異なる設問に対して提出されたメソッド” という仮定の下で収集されている．このとき，コードクローンに比べて非コードクローンの数が非常に多くなる．そのため，本評価実験では非コードクローンの数をコードクローンと同数になるように調整した．また，オープンジャッジの特性上，全てのメソッドに入力を受け付けるためのコード片や設問に対する解答を出力するためのコード片が含まれる．これ

らのコード片は本評価実験では意図していないコードクローンとみなし，データセットに利用する回答ソースコードからこれらのコード片を削除した．

SemanticCloneBench（以降, SCB）[11] は, Al-omari らによって提案された意味的コードクローンのベンチマークであり，Stack Overflow(注3)のコード片を用いて構築されている．このベンチマークは，“同じ質問に対する回答コード片は互いに意味的コードクローンである” という仮定の下構築されている．このベンチマークには 1000 個の意味的コードクローンが含まれていたが，そのうち 139 個に Javalang(注4)で構文解析できないメソッドが含まれていたため，残りの 861 個を利用した．更に本研究では，861 個の非コードクローンを作成した．具体的には，SCB に含まれるメソッドからランダムに二つ選択し，それらがコードクローンとして SCB に登録されていなければ，それらの二つのメソッドを非コードクローンとしてデータセットに追加した．

SeSaMe [12] は Kamp らによって提案された意味的コードクローンのデータセットで，オープンソースソフトウェアのメソッドが用いられている．このデータセットは，“JavaDoc の文章が類似しているメソッドは意味的コードクローンである” という仮定に基づいて構築されており，900 個の意味的コードクローンが含まれている．しかし，このデータセットには，API を呼び出すだけのメソッド等，短いメソッドが多く含まれていた．また，JavaDoc の文章全体の意味は異なっていても同じ単語が文章に使われているために，実際にはコードクローンではないがコードクローンとしてデータセットに含まれているペアが存在していた．そのようなペアを取り除くため，本研究では SeSaMe データセットを目視確認した．その結果，114 個のメソッドペアが実際のコードクローンであることを確認した．そして，SCB と同様の方法で 114 個の非コードクローンを作成した．

CodeSearchNet（以降，CSN）[19] は，コード片に特化した検索エンジンである．本評価実験では，“CSN でコード検索を行うとき，同じ検索クエリで検索可能なコード片は意味的コードクローンである” という仮定の下，485100 個のコードクローン候補ペアを収集した．その中から短いメソッドを含むペアを削除したあと，コードクローン候補ペアを手動で確認し，コードクローンのラベリングを行った．ただし，485100 個全てを目視確認することは困難であるため，コードクローンであると判断したペアが 300 個現れるまでランダムな順で目視確認を行い，見つけた 300 個のペアをコードクローンとした．そして，SCB・SeSaMe と同様に 300 個の非コードクローンを作成した．このとき，300 個の非コードクローンペアの中に 485100 個のコードクローン候補ペアは含まれないようにした．

4.3 評価対象モデル

本章では評価対象として選択した四つの深層学習モデルについて説明する．本評価実験で用いるデータセットにはコンパイルが難しいソースコードが含まれるため，本評価実験ではソースコードをモデルに入力する際にコンパイルを必要としない深層学習モデルを選択した．

4.3.1 ASTNN

ASTNN [1] は，ソースコードの AST をステートメントレベルに分割してそれぞれベクトル化してから Bi-directional Gated Recurrent Unit (Bi-GRU) [20] にステートメントベクトルの深さ優先探索順列を入力することでコード片をベクトル化し，Siamese モデルを用いて意味的コードクローン検出を行う深層学習モデルである．著者らによってモデルのソースコードやデータセットが公開されており，コードクローン検出対象のコード片のコンパイルが不要なため，ASTNN を本評価実験に用いる．

4.3.2 LSTM・Bi-LSTM

Long Short-Term Memory（以降，LSTM）[21] は時系列データに対応可能な深層学習モデルの一つである．LSTM は前から順に系列を読み込むのに対し，Bi-directional Long Short-Term Memory（以降，Bi-LSTM）は前からの読み込みと同時に後ろからも系列を読み込む．ソースコードはトークン列などの系列データとして表現可能なため，LSTM と Bi-LSTM はコードクローン検出手法に利用されている [1], [5]．よって，これらの深層学習モデルを，本評価実験における Siamese モデルの埋め込み層に用いる．また本評価実験では，目的変数に使用する AST 類似度との親和性を考え，トークン列ではなく，AST ノードの深さ優先探索順列（先行順）を説明変数に使用する．

本手法では最初に，ASTNN の構文解析器を利用してソースコードの構文解析を行い，AST ノードの深さ優先探索順列（先行順）を作成する．2 番目に，作成し

(注3)：https://stackoverflow.com/
(注4)：https://github.com/c2nes/javalang

表3 実験結果

深層学習モデル	データセット	分類層	適合率	再現率	F 値	AUC	深層学習モデル	データセット	分類層	適合率	再現率	F 値	AUC
ASTNN	GCJ	2 値分類	0.521	0.314	0.392	0.513	Bi-LSTM	GCJ	2 値分類	0.521	0.687	0.593	0.537
		回帰	0.575	0.877	0.694	0.724			回帰	0.508	0.986	0.671	0.671
	SCB	2 値分類	0.669	0.285	0.399	0.667		SCB	2 値分類	0.644	0.582	0.611	0.672
		回帰	0.706	0.820	0.759	0.832			回帰	0.606	0.904	0.726	0.810
	SeSaMe	2 値分類	0.974	0.325	0.487	0.746		SeSaMe	2 値分類	0.741	0.351	0.476	0.637
		回帰	0.791	0.763	0.777	0.844			回帰	0.622	0.807	0.702	0.753
	CSN	2 値分類	0.843	0.197	0.319	0.711		CSN	2 値分類	0.689	0.280	0.398	0.628
		回帰	0.745	0.643	0.691	0.785			回帰	0.629	0.683	0.655	0.703
LSTM	GCJ	2 値分類	0.506	0.936	0.657	0.511	FNN	GCJ	2 値分類	0.562	0.922	0.698	0.717
		回帰	0.517	0.979	0.676	0.743			回帰	0.690	0.606	0.645	0.725
	SCB	2 値分類	0.659	0.659	0.659	0.718		SCB	2 値分類	0.545	0.931	0.685	0.757
		回帰	0.597	0.914	0.722	0.842			回帰	0.575	0.804	0.670	0.687
	SeSaMe	2 値分類	0.682	0.509	0.583	0.689		SeSaMe	2 値分類	0.528	0.895	0.664	0.715
		回帰	0.662	0.807	0.727	0.804			回帰	0.556	0.789	0.652	0.634
	CSN	2 値分類	0.661	0.383	0.485	0.631		CSN	2 値分類	0.519	0.837	0.640	0.587
		回帰	0.671	0.823	0.740	0.788			回帰	0.498	0.723	0.590	0.529

た順列の順番に従って AST ノードを一つずつ LSTM または Bi-LSTM に入力する．その結果，ソースコードの埋め込みベクトルが出力される．その埋め込みベクトルを用いて，Siamese モデルによる意味的コードクローン検出を行う．

4.3.3 FNN

Feedforward Neural Networks（以降，FNN）[22] は，ネットワークにループ構造が含まれない標準的なニューラルネットワークである．ソースコードメトリクスを並べるなどの方法でソースコードのベクトル化を行うことで，FNN をコードクローン検出に利用することができる [2], [3], [6], [23]．本評価実験では，ソースコードのコンパイルが不要な既存手法の一つである CLCDSA [23] を選択し，CLCDSA で利用されているソースコードメトリクスと FNN モデルを再現して実験を行う．

4.4 回帰モデルのしきい値

3.1 で説明したとおり，回帰モデルの出力は実数になるため，しきい値を設定する必要がある．そのため事前実験として，BCB のソースコードを学習させた ASTNN の回帰モデルを BCB のソースコードで評価した．その結果，しきい値が 0.1 のときに最も F 値が高くなった．そのため，本評価実験における回帰モデルではしきい値を 0.1 に設定した．

4.5 実験結果

実験結果を表 3 に示す．この表から分かるように，2 値分類モデルから回帰モデルに変更すると，ASTNN，LSTM，Bi-LSTM の場合は全てのデータセットに対して F 値と AUC が共に高くなった．ASTNN の場合は平均で F 値が 0.341，AUC が 0.253 だけ上昇し，LSTM の場合は平均で F 値が 0.120，AUC が 0.157 だけ上昇し，Bi-LSTM の場合は平均で F 値が 0.169，AUC が 0.116 だけ上昇した．FNN の場合は GCJ に対しては F 値と AUC が高くなったが，GCJ 以外のデータセットに対しては F 値と AUC が低くなった．平均すると F 値が 0.033，AUC が 0.050 だけ低下した．以上のように，2 値分類モデルから回帰モデルに変更することによって，四つ中三つの深層学習モデルのコードクローン検出精度が向上することが分かった．

4.6 考察

評価実験では 2 値分類モデルと回帰モデルのコードクローン検出精度比較を行った．その結果，FNN 以外の深層学習モデルにおいてコードクローン検出精度が向上することが確認できた．FNN で精度が向上しなかったのは，入力としてモデルに与えたソースコード構造に関する情報が，他のモデルと比べて少ないためだと考えられる．FNN 以外のモデルは AST ノードの深さ優先探索順列を入力として与えられ，AST ノードの順序関係を学習しているのに対し，FNN にはソースコードメトリクスのベクトルを入力として与えており，ソースコードの構造を学習していない．そのため，回帰モデルで類似度スコアを予測することが難しく，精度がほとんど変わらなかったと考えられる．

次に，回帰モデルを用いることによって学習に利用できるようになった AST 類似度の影響を調べるために，評価データにおけるコードクローンと非コードクローンの AST 類似度と，回帰モデルに変更した ASTNN が出力した類似度スコアの分布を図 3 に示す．各図の横軸はコード片ペアの AST 類似度 (AST Similarity) であり，縦軸は ASTNN が出力した類似度スコア (score) である．そして，コード片ペアごとの AST 類似度と ASTNN による類似度スコアを黒点でプロットし，回

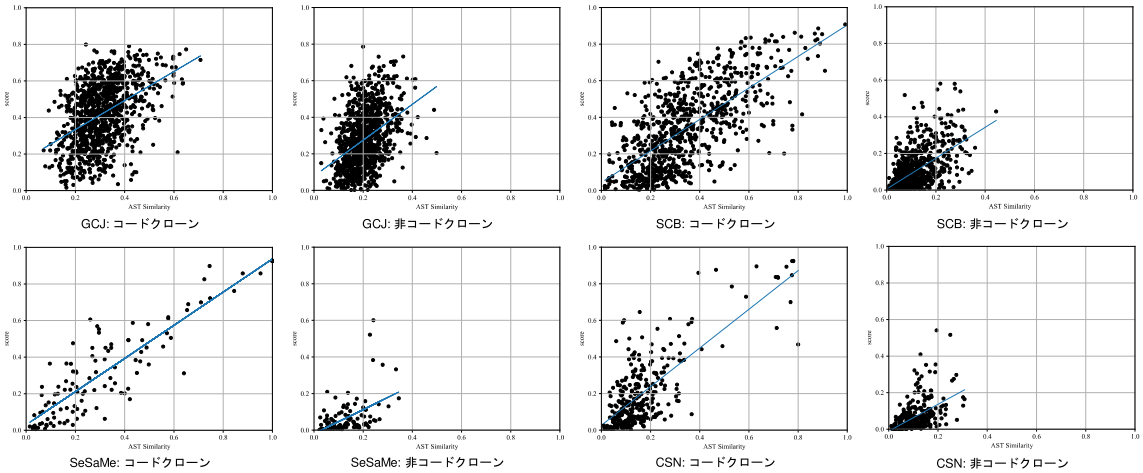

図 3　AST 類似度・予測スコアの散布図と回帰直線

帰直線を青線で引いている．ただし GCJ のコード片ペアは非常に数が多いため，コードクローンと非コードクローンをそれぞれ 1000 ペアだけランダムに抽出しプロットしている．この図を見ると，AST 類似度と類似度スコアに相関があり，ASTNN は入力されたコード片ペアの AST 類似度に近い値を予測していることが分かる．また，図 3 から分かるように，多くのペアの AST 類似度は 0.0 から 0.4 の範囲に含まれている．そして，この範囲では全てのデータセットにおいてコードクローンの回帰直線が非コードクローンの回帰直線の上部に描かれている．これは，同じ AST 類似度のコードクローンと非コードクローンをそれぞれ入力した場合に，回帰モデルは非コードクローンよりもコードクローンに対して高い類似度スコアを出力する傾向があることを示している．したがってこの結果から，本評価実験における回帰モデルは AST の類似度だけでなく意味的コードクローンの特徴を類似度スコアに反映する能力があると考えられる．

また，図 3 から，SCB と SeSaMe データセットには，AST 類似度が高いコードクローンが比較的多く含まれていることが分かった．具体的には，AST 類似度が 0.5 以上のコードクローンが，SCB に 146 個，SeSaMe データセットに 20 個含まれていた．本論文では，意味的コードクローン検出手法，すなわち構文的な類似度が低いコードクローンを検出可能な手法に注目しているため，AST 類似度が低いコードクローンに対する検出精度を調べることが重要である．そのため，SCB と SeSaMe データセットから，AST 類似度が 0.5 以上のコードクローンを取り除き，コードクローン検出精度の評価実験を行った．その実験結果を表 4 に示す．表 4 から分かるとおり，FNN 以外の場合で，提案手法である回帰モデルの方が 2 値分類モデルよりも F 値と AUC は高かった．この結果から，検出対象を意味的コードクローンに限定した場合でも，提案手法を適用することで FNN 以外の場合に深層学習モデルのコードクローン検出精度が向上することが確認できた．

表 4　AST 類似度が 0.5 以上のコードクローンを取り除いた場合の実験結果

深層学習モデル	データセット	分類層	適合率	再現率	F 値	AUC
ASTNN	SCB	2 値分類	0.582	0.231	0.331	0.634
		回帰	0.649	0.782	0.709	0.802
	SeSaMe	2 値分類	0.957	0.234	0.376	0.700
		回帰	0.744	0.713	0.728	0.811
LSTM	SCB	2 値分類	0.598	0.606	0.602	0.685
		回帰	0.547	0.896	0.679	0.814
	SeSaMe	2 値分類	0.585	0.404	0.478	0.626
		回帰	0.605	0.766	0.676	0.762
BiLSTM	SCB	2 値分類	0.575	0.521	0.547	0.634
		回帰	0.556	0.883	0.683	0.780
	SeSaMe	2 値分類	0.622	0.245	0.351	0.572
		回帰	0.562	0.766	0.649	0.702
FNN	SCB	2 値分類	0.492	0.918	0.640	0.732
		回帰	0.519	0.769	0.619	0.656
	SeSaMe	2 値分類	0.474	0.872	0.614	0.661
		回帰	0.493	0.745	0.593	0.570

また，本論文では，評価データセットのコードクローン数と非コードクローン数を揃えて評価実験を行った．しかし，実際のソースコードに対してコードクローン検出を行う場合，メソッドのペアがコードクローンかどうかを総当たりで調査する必要がある．そのため，評価データセットのコードクローンを構成するメソッドを利用し，メソッドのペアを総当たりで調

査するための現実に即した評価データセットを作成した．その内訳を表 5 に示す．表 5 の “メソッド数” は，評価データセットのコードクローンを構成するメソッドの種類数に等しい．SCB では，コードクローンを構成するメソッドに重複がないため，メソッド数はコードクローン数の 2 倍である．その他のデータセットでは，一つのメソッドが複数のコードクローンを構成するため，コードクローン数の 2 倍よりもメソッド数が少ない．また，CSN は，542 個のメソッドの全てのペアを作成したとき，評価データセットでラベリングされていた 300 個のコードクローンの他に **4.2.2** で説明したコードクローン候補が約 2000 個存在した．そのため，コードクローン候補の目視確認を行った．その結果，全体のコードクローン数は 697 個になった．

現実に即したデータセットでの実験結果を表 6 に示す．適合率が 0.001 を下回る場合があったため，表 6 の数値は小数第 4 位まで表記している．**4.2** のデータセットと比べて非コードクローン数が大幅に増えたため，全ての場合で適合率と F 値が低下した．CSN 以外のデータセットは含まれているコードクローンに変化がないため，再現率は変わらなかった．一方，CSN データセットに新たに追加されたコードクローンを高精度で検出することができたため，CSN データセットに対する再現率は上昇した．また，SeSaMe に対する ASTNN の実験結果など，表 3 において 2 値分類モデルの適合率が高いデータセットと分類層の組合せの一部で，2 値分類モデルと回帰モデルの F 値の優劣が逆転した．しかし，AUC は表 3 と同様に FNN 以外の場合で回帰モデルの方が高かった．AUC が高いという結果から，しきい値（**4.4** 参照）を評価データセットごとに適切に設定すれば，FNN 以外の場合では回帰モデルの方が高精度なコードクローン検出を行うことができる．実際に，SeSaMe に対する ASTNN の実験結果に関して，F 値が最も高くなるしきい値について調査した．その結果，2 値分類モデルはしきい値が 0.8 のときに F 値が 0.0952 で最も高くなったのに対し，回帰モデルはしきい値が 0.45 のときに F 値が 0.2625 で最も高くなり，回帰モデルの F 値が 2 値分類モデルの F 値を上回ったことを確認した．ただし，本論文では意味的コードクローン検出モデルの汎化性能の評価を目的としているため，**4.4** のように学習データセットの検証結果に基づいてしきい値を決定する必要があり，評価データセットごとにしきい値を変えることはできない．よって，評価データセットごとに適切なしきい値が変わることなく一定になるように学習を行うことが，提案手法の課題の一つであることが分かった．

表 5　現実に即した評価データセットの内訳

	メソッド数	コードクローン	非コードクローン
GCJ	1663	274048	1107905
SCB	1722	861	1480920
SeSaMe	194	114	18607
CSN	542	697	145914

5. 関 連 研 究

現在まで，深層学習を用いた意味的コードクローン検出に関する研究が多く存在する．Deepsim [2] は，制御/データフローグラフの情報を用いてコード片をベクトル化し，それを順伝播型ニューラルネットワークを用いて解析し，意味的コードクローン検出を行う．CDLH [5] は，AST に対して tree-LSTM を適用することで抽象構文木を用いたコードクローン検出を行う．CCLearner [6] は，Java のメソッドを，八つのカテゴリーからなるトークン発生頻度ベクトルとして表現し，それを順伝播型ニューラルネットワークを用

表 6　現実に即したデータセットでの実験結果

深層学習モデル	データセット	分類層	適合率	再現率	F 値	AUC	深層学習モデル	データセット	分類層	適合率	再現率	F 値	AUC
ASTNN	GCJ	2 値分類	0.1370	0.3141	0.1908	0.3494	Bi-LSTM	GCJ	2 値分類	0.2145	0.6874	0.3270	0.5459
		回帰	0.2498	0.8771	0.3889	0.7228			回帰	0.2037	0.9856	0.3377	0.6943
	SCB	2 値分類	0.0013	0.2846	0.0026	0.6787		SCB	2 値分類	0.0011	0.5819	0.0022	0.6789
		回帰	0.0014	0.8200	0.0028	0.8285			回帰	0.0010	0.9036	0.0019	0.8220
	SeSaMe	2 値分類	0.0247	0.3246	0.0458	0.6834		SeSaMe	2 値分類	0.0117	0.3509	0.0226	0.6254
		回帰	0.0148	0.7632	0.0290	0.7988			回帰	0.0095	0.8070	0.0187	0.7640
	CSN	2 値分類	0.0272	0.3257	0.0503	0.7843		CSN	2 値分類	0.0175	0.4534	0.0336	0.7387
		回帰	0.0133	0.7604	0.0262	0.8263			回帰	0.0105	0.8077	0.0208	0.8197
LSTM	GCJ	2 値分類	0.2019	0.9356	0.3321	0.5067	FNN	GCJ	2 値分類	0.2407	0.9221	0.3817	0.7172
		回帰	0.2092	0.9785	0.3448	0.7412			回帰	0.3547	0.6056	0.4474	0.7256
	SCB	2 値分類	0.0012	0.6585	0.0024	0.7255		SCB	2 値分類	0.0007	0.9314	0.0014	0.7457
		回帰	0.0009	0.9141	0.0017	0.8436			回帰	0.0008	0.8037	0.0015	0.6825
	SeSaMe	2 値分類	0.0119	0.5088	0.0233	0.6683		SeSaMe	2 値分類	0.0066	0.8947	0.0132	0.7101
		回帰	0.0097	0.8070	0.0192	0.7758			回帰	0.0073	0.7894	0.0145	0.6536
	CSN	2 値分類	0.0154	0.5538	0.0299	0.7638		CSN	2 値分類	0.0053	0.8451	0.0106	0.6404
		回帰	0.0087	0.9082	0.0173	0.8602			回帰	0.0056	0.7475	0.0110	0.6353

いて解析し，コードクローン検出を行う．Oreo [3] や CLCDSA [23] は，Java のメソッドをソースコードメトリクスのベクトルで表現し，それを Siamese モデルを用いて解析することで，意味的コードクローン検出を行う．TBCCD [24] は，トークン情報をノードに付与した AST をベクトル化し，Siamese モデルを用いて解析を行うことで意味的コードクローン検出を行う．FCCA [25] は，一つのソースコードから三つのソースコード表現（トークン列，AST，制御フローグラフ）を生成し，三つの深層学習モデル（LSTM，Tree-LSTM，グラフ畳み込みネットワーク）を使ってそれぞれのソースコード表現を学習することで，意味的コードクローン検出を行う．この手法は，一般的な既存手法で用いられている 2 値分類モデルではなく，非コードクローン・タイプ 1 クローン・タイプ 2 クローン・タイプ 3 クローン・タイプ 4 クローンの 5 通りで分類を行う 5 値分類モデルであることが特徴の一つである．以上の研究では，同じデータセットから学習データと評価データを作成しているため，学習データと評価データの間に依存関係が生じており，過学習が発生している可能性がある．一方，本評価実験では BCB から学習データを作成し，BCB 以外のデータセットから評価データを作成しているため，学習データと評価データの間に依存関係は生じておらず，深層学習モデルの汎化性能をより正確に評価することができる．

6. むすび

本研究では，回帰モデルを用いたコードクローン検出手法を提案し，その汎化性能について評価した．提案手法では，深層学習によるコードクローン検出の既存手法で一般的に用いられている 2 値分類モデルの代わりに回帰モデルを用いることで，コードクローンの AST 類似度を学習に利用した．評価実験では，学習と評価に異なるデータセットを用いて依存関係を取り除き，適合率・再現率・F 値・AUC を用いて深層学習モデルのコードクローン検出精度を評価した．その結果，回帰モデルを用いると四つ中三つの深層学習モデルでコードクローン検出精度が向上することを確認した．

今後の課題として以下の点を挙げる．

- 本論文の評価実験で用いたもの以外の既存手法や深層学習モデルに対しても，提案手法が意味的コードクローン検出精度の向上に有効かを調査する．
- 目的変数ではなく説明変数として AST 類似度を用いる手法が，意味的コードクローン検出精度の向上に有効かを調査する．
- データセットごとに適切なしきい値が一定の値になるように深層学習モデルの学習を行う方法について調査する．

文　献

[1] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu, “A novel neural source code representation based on abstract syntax tree,” Proc. ICSE 2019, pp.783–794, Montréal, QC, Canada, May 2019. DOI:10.1109/ICSE.2019.00086

[2] G. Zhao and J. Huang, “Deepsim: deep learning code functional similarity,” Proc. FSE 2018, pp.141–151, Lake Buena Vista, FL, USA, Nov. 2018. DOI:10.1145/3236024.3236068

[3] V. Saini, F. Farmahinifarahani, Y. Lu, P. Baldi, and C.V. Lopes, “Oreo: Detection of clones in the twilight zone,” Proc. ESEC/FSE 2018, pp.354–365, Lake Buena Vista, FL, USA, Oct. 2018. DOI:10.1145/3236024.3236026

[4] M. White, M. Tufano, C. Vendome, and D. Poshyvanyk, “Deep learning code fragments for code clone detection,” Proc. ASE 2016, pp.87–98, Singapore, Singapore, Sept. 2016. DOI:10.1145/2970276.2970326

[5] H. Wei and M. Li, “Supervised deep features for software functional clone detection by exploiting lexical and syntactical information in source code,” Proc. IJCAI 2017, pp.3034–3040, Melbourne, Australia, 2017. DOI:10.24963/ijcai.2017/423

[6] L. Li, H. Feng, W. Zhuang, N. Meng, and B. Ryder, “Cclearner: A deep learning-based clone detection approach,” Proc. ICSME 2017, pp.249–260, Shanghai, China, Sept. 2017. DOI:10.1109/ICSME.2017.46

[7] J. Svajlenko, J.F. Islam, I. Keivanloo, C.K. Roy, and M.M. Mia, “Towards a big data curated benchmark of inter-project code clones,” Proc. ICSME 2014, pp.476–480, Victoria, BC, Canada, Sept. 2014. DOI:10.1109/ICSME.2014.77

[8] 肥後芳樹，楠本真二，井上克郎，“コードクローン検出とその関連技術，” 信学論（D），vol.J91-D, no.6, pp.1465–1481, June 2008.

[9] I.D. Baxter, A. Yahin, L. Moura, M.S. Anna, and L. Bier, “Clone detection using abstract syntax trees,” Proc. ICSM 1998, pp.368–377, Bethesda, MD, USA, March 1998. DOI:10.5555/850947.853341

[10] C.K. Roy, J.R. Cordy, and R. Koschke, “Comparison and evaluation of code clone detection techniques and tools: A qualitative approach,” Science of Computer Programming, vol.74, no.7, pp.470–495, May 2009. DOI:10.1016/j.scico.2009.02.007

[11] F. Al-Omari, C.K. Roy, and T. Chen, “Semanticclonebench: A semantic code clone benchmark using crowd-source knowledge,” Proc. IWSC 2020, pp.57–63, London, ON, Canada, Feb. 2020. DOI:10.1109/IWSC50091.2020.9047643

[12] M. Kamp, P. Kreutzer, and M. Philippsen, “Sesame: A data set of semantically similar java methods,” Proc. MSR 2019, pp.529–533, Montreal, QC, Canada, Canada, May 2019. DOI:10.1109/MSR.2019.00079

[13] J. Bromley, J.W. Bentz, L. Bottou, I. Guyon, Y. Lecun, E. Sackinger, and R. Shah, “Signature verification using a ‘siamese’ time delay neural network,” Int. J. Pattern Recognit. Artif. Intell., vol.7, no.4, pp.669–688, 1993. DOI:10.1142/S0218001493000339

[14] K. Zhang and D. Shasha, “Simple fast algorithms for the editing distance between trees and related problems,” SIAM J. Comput., vol.18, no.6, pp.1245–1262, 1989. DOI:10.1137/0218082

[15] M. Hashimoto and A. Mori, “Diff/TS: A tool for fine-grained structural change analysis,” Proc. WCRE 2008, pp.279–288, Washington, DC, USA, 2008. DOI:10.1109/WCRE.2008.44

[16] A.P. Bradley, “The use of the area under the roc curve in the evaluation of machine learning algorithms,” Pattern Recognit., vol.30, no.7, pp.1145–1159, 1997. DOI:10.1016/S0031-3203(96)00142-2

[17] K. Kim, D. Kim, T.F. Bissyandé, E. Choi, L. Li, J. Klein, and Y.L. Traon, “FaCoY: A code-to-code search engine,” Proc. ICSE 2018, pp.946–957, Gothenburg, Sweden, 2018. DOI:10.1145/3180155.3180187

[18] Y. Gao, Z. Wang, S. Liu, L. Yang, W. Sang, and Y. Cai, “Teccd: A tree embedding approach for code clone detection,” Proc. ICSME 2019, pp.145–156, Cleveland, OH, USA, Oct. 2019. DOI:10.1109/ICSME.2019.00025

[19] H. Husain, H.-H. Wu, T. Gazit, M. Allamanis, and M. Brockschmidt, “Codesearchnet challenge: Evaluating the state of semantic code search,” 2019. http://arxiv.org/abs/1909.09436

[20] D. Tang, B. Qin, and T. Liu, “Document modeling with gated recurrent neural network for sentiment classification,” Proc. EMNLP 2015, pp.1422–1432, Lisbon, Portugal, Sept. 2015. DOI:10.18653/v1/D15-1167

[21] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol.9, no.8, pp.1735–1780, 1997. DOI:10.1162/neco.1997.9.8.1735

[22] J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural Netw., vol.61, pp.85–117, 2015. DOI:10.1016/j.neunet.2014.09.003

[23] K.W. Nafi, T.S. Kar, B. Roy, C.K. Roy, and K.A. Schneider, “Clcdsa: Cross language code clone detection using syntactical features and api documentation,” Proc. ASE 2019, pp.1026–1037, San Diego, CA, USA, Nov. 2019. DOI:10.1109/ASE.2019.00099

[24] H. Yu, W. Lam, L. Chen, G. Li, T. Xie, and Q. Wang, “Neural detection of semantic code clones via tree-based convolution,” Proc. ICPC 2019, p.70–80, Montreal, Quebec, Canada, 2019. DOI:10.1109/ICPC.2019.00021

[25] W. Hua, Y. Sui, Y. Wan, G. Liu, and G. Xu, “Fcca: Hybrid code representation for functional clone detection using attention networks,” IEEE Trans. Reliab., pp.1–15, 2020. DOI:10.1109/TR.2020.3001918

（2020 年 11 月 18 日受付，2021 年 2 月 26 日再受付，
5 月 17 日早期公開）

藤原　裕士

平 29 大阪大学基礎工学部情報科学科卒．令和元年度大阪大学大学院情報科学研究科博士前期課程了．現在，大阪大学大学院情報科学研究科博士後期課程に在学中．令和元年より，産業技術総合研究所リサーチアシスタントに従事．深層学習を用いたコードクローン分析手法に関する研究に従事．

森　　彰

1995 京都大学大学院工学研究科情報工学専攻博士課程了（博士（工学））．英国オックスフォード大学訪問研究員，米国カリフォルニア大学サンディエゴ校ポスドク，北陸先端科学技術大学院大学助手を経て，2001 産業技術総合研究所入所．現在同所研究チーム長．ソフトウェア工学及びコンピューターセキュリティに関わるプログラム解析技術の研究に従事．

井上　克郎　（正員：フェロー）

昭 59 大阪大学大学院基礎工学研究科博士後期課程了（工学博士）．同年大阪大学基礎工学部情報工学科助手．昭 59〜61，ハワイ大学マノア校コンピュータサイエンス学科助教授．平 3 大阪大学基礎工学部助教授．平 7 同学部教授．平 14 より大阪大学大学院情報科学研究科教授．平 28 より産業技術総合研究所サイバーフィジカルセキュリティー研究センター特定フェロー．ソフトウェア工学，特にコードクローンやコード検索などのプログラム分析や再利用技術の研究に従事．