解　説

コードクローン検索手法の調査

崔 恩瀞　藤原 裕士　吉田 則裕　水野 修

ソフトウェアの品質と生産性を向上させるために開発者は頻繁にコード検索を行う．開発者のコード検索作業を支援するため，クエリとして入力されたコード片のコードクローン (類似コード片) を検索する手法が多く提案されている．本稿では，既存のコードクローン検索手法で使われているコードクローン検出技術と評価指標を調査する．まず，コードクローン検索手法の中で使用されている技術を分類する．次に，コードクローン検索研究で使用されている評価指標を分析し，最後に，今後行われる研究の方向性について考察を行う．

To improve the quality and productivity of software, developers frequently perform code searches. To support this task, many previous research studies have proposed code clone search approaches that search a set of code clones (similar code fragments) of an input code fragment. In this survey paper, we investigate code clone detection techniques and evaluation metrics used in the previous studies. In detail, at first, we categorize code clone detection techniques and then introduce code search approaches in each category. Then, we further elaborate evaluation metrics used in previous studies. Finally, we discuss possible future research directions on code clone search.

1　はじめに

ソフトウェアを効率よく開発するために，開発者は頻繁にコード検索を行う [46]．開発者のコード検索作業を支援するために，現在まで様々なコード検索手法が提案されている．コード検索手法は，開発者が与えた情報に基づいて開発者の要求にあうコード片を検索することを目的としている．コード検索手法は入力として，自然文 [12] [25]，コード片 [27] [53]，入出力の例 [47] [48] などを受け取ることができる．多数のコード検索手法はテキスト形式の自然文をクエリとして使用して開発者が入力した自然文と一致するコード片を検索している [2] [12] [25]．しかし，自然文によるコード検索手法は開発者が自分の要求を自然言語で正確に表現することは難しいため，開発者の要求に合わないコード片を検索する可能性が高い．その反面，入出力の例やコード片などを入力として使用するコード検索手法では，開発者が検索したいソースコードに対する要求を具体的に示すことが出来るため，開発者の要求にあうコード片が検索される可能性が高いと考えられる．例えば，入出力の例に基づいたコード検索手法では，開発者が提示したコード片の入出力例に適合するコード片を検索する．また，コードクローン検索手法では，開発者が提示したコード片とコードクローン (ソースコード中に存在する互いに一致または類似した部分を持つコード片 [18]) 関係であるコード片を見つける．このように，開発者はコードクローン検索手法を用いて自分の要求にあうコード片を検索できるため，これまでに多様なコードクローン検索手法が提案されている [23] [27] [53].

コードクローン検索手法では入力として与えられたコード片とコードクローン関係であるコード片の集合を検索対象のソースコード中から見つける．開発者はコードクローン検索手法の検索結果を用いて，

A Survey of Approaches for Code Clone Search.
Eunjong Choi, Osamu Mizuno, 京都工芸繊維大学, Kyoto Institute of Technology.
Yuji Fujiwara, 大阪大学, Osaka University.
Norihiro Yoshida, 名古屋大学, Nagoya University.
コンピュータソフトウェア, Vol.39, No.3 (2022), pp.47–59.
[解説論文] 2021 年 4 月 16 日受付.

既存のソースコードの再利用，パッチ作成やリファクタリングなど様々な開発タスクを効率よく行うことができ，ソフトウェアの品質や生産性の向上が期待できる[29]．例えば，ソートアルゴリズムを実装したコード片をコードクローン検索手法に入力することで，そのソートアルゴリズムもしくは類似したアルゴリズムを実装しているコード片が検索される．従って，開発者は検索されたコード片の中から再利用対象のコード片を効率的に取得することができる．

Liu らはコード検索ツールに関する系統的レビューを実施し，その結果を arXiv 上に公開した[26]．彼らは，2002 年から 2019 年までに出版された 81 編の文献で紹介されているコード検索ツールをコード検索作業に基づいて分類した．また，コード検索ツールの研究動向を明らかにし，研究の課題と今後の方向性を議論した．しかし，彼らの調査はコード検索ツール分類に不正確な部分があり，かつコードクローン検出技術を用いた検索手法に関して詳細な解説を行っていない (2.1 節参照)．

そこで，Liu らの調査における問題を緩和するため，本稿では彼らの調査結果に基づいて現在まで提案されているコードクローン検索手法を中心に紹介を行う．現在まで様々なコードクローンに関するサーベイが行われている (5 章参照)．しかし，我々が知る限りコードクローン検索に関する文献のみ着目したサーベイは行われていないが，Liu らの論文でコード検索手法の一部としてコードクローン検索に関する文献が紹介されている．従って，本稿は Liu らの論文に基づいてコードクローン検索に関する文献を調査する．具体的に，本稿はコードクローン検索に関する 10 編の文献を調査し，コードクローン検索手法の中で使用されているコードクローン検出技術と評価指標を明らかにした．現在，コードクローン検索手法に関する研究は早期段階であり今後より多くの手法の提案が期待できる．従って，本調査でコードクローン検索手法で使われているコードクローン検出技術と評価指標を整理することで，実務者や研究者にとって本稿の調査が参考になるとともに，研究者にとってはコードクローン検索に関する今後の研究方針を考える際に参考になると考えられる．

本稿では，まず，2 章において，本稿の背景を説明し，3 章では，コードクローン検索手法の紹介を行う．次に，4 章では今後行われる研究の方向性について議論し，5 章では関連研究として既存のコードクローンに関するサーベイを紹介する．最後に，6 章において本稿のまとめを述べる．

2 背景

2.1 既存のコード検索に関する調査論文

Liu らはコード検索ツール研究に関する系統的レビューを実施し，その結果を arXiv 上に公開した[26]．具体的に，広く使われている電子データベース ACM Digital Library, IEEE Xplore, ISI Web of Science 上で検索キーワード “code search”，“code retrieval” を用いて文献のタイトル，概要，キーワードを検索し，既存のコード検索ツールに関する文献を見つけた．その後に．以下の適格規準 (Inclusion Criteria) と除外規準 (Exclusion Criteria) を適用して調査対象の文献を決定した．

IC： 英語で書かれていること．

IC： コード検索タスクに対応するツールが少なくとも 1 つ含まれていること．

IC： 会議録または雑誌で出版された査読付きフル研究論文であること．

EC： 基調講演の記録や灰色文献 (Grey literature) は除外する．

EC： 国際会議論文と国際会議論文の発展版として発行されているジャーナル論文がある場合，国際会議論文は除外する．

EC： 新しいコード検索ツールを提案しているが，その性能を評価していない研究は除外する．

EC： 既存のコード検索ツールを他のソフトウェア開発上のタスク (例：バグの特定やプログラムの修復) に適用した研究は除外する．

その結果，調査対象として 81 編の文献が選択された．また，既存のコード検索ツールをコード検索作業に基づいて以下の 7 つに分類した．

文字列に基づいたコード検索： 開発者が自由に入力した文字列の内容に最も適合するコード片を検索する．

コードクローン検索： 入力されたコード片のコードクローンを検索する．

入出力の例に基づいたコード検索： 入力された API の入出力例に適合するコード片を検索する．

API に基づいたコード検索： 与えられた API 名に基づいて API の例を検索する．

バイナリコード検索： 入力されたバイナリコードに対して，同じバイナリコードを他の形式でコンパイルしたものを検索する．

UI(ユーザインタフェース) に基づいたコード検索： 入力された開発者の要求 (例：UI スケッチ) に合う UI 実装コードを検索する．

プログラミングビデオ検索： 文字列のクエリを用いて YouTube などのプログラミングビデオから関連するコード片を検索する．

また，彼らの調査によって以下のようなコード検索に関する研究動向が明らかになった．

- コード検索ツールに関する研究が，2002 年にスタートし 2019 年には出版のピークを迎えている
- 近頃，深層学習モデルが注目されている
- 既存研究で転置索引が広く使用されている
- 評価指標として MRR(Mean Reciprocal Rank) や NDCG (Normalized Discounted Cumulative Gain) などランキングに関する評価指標が頻繁に使用されている

また，これらの調査結果に基づいてコード検索ツールの研究課題と今後の方向性を議論した．しかし，彼らの調査は 1 章で説明した通り，以下の問題がある．

問題 1： コードクローン検出技術とコードクローン検索技術を区別せず，コードクローン検出技術をコードクローン検索技術として分類している部分がある．

問題 2： コード検索技術を概観することを目的としているため，コードクローン検出技術を用いた検索手法に関して詳細な解説を行っていない．そのため，コードクローン検出技術を用いた検索手法の研究動向を把握することが難しい．

2.2 コードクローン検索

本節では，まず，コードクローン，コードクローン検出とコードクローン検索について説明する．コードクローンとは，ソースコード中に存在する互いに一致または類似した部分を持つコード片である．また，コードクローン関係である任意のコード片は反射，推移，対称律を満たし，同値関係である．コードクローン検出では入力として与えられたソースコードからコードクローン関係である全てのコード片の対，あるいは全ての集合を見つける．入力ソースコードが大規模の場合，コードクローン検出に膨大な時間的・空間的コストを必要とする．近年，スケーラビリティの向上をもたらすために同値関係に基づいてコードクローン検出を近似的に行う手法が多く提案されている [19] [52]．このような手法では，同値関係を必ずしも維持しておらず，そのようなコード片でもコードクローン関係にあるとみなしている．コードクローン検索では，クエリ q が入力として与えられた場合，検索対象であるコード片の集合 T から q とコードクローン関係であるコード片の集合 C を見つけて提示する．ここで，クエリ q として構文的に完全もしくは不完全な 1 つのコード片 (例：メソッド，コードブロック) を使用する．コードクローン検索では，C を見つけるために，クエリ q と T に含まれている全てのコード片の間の類似度を計算する必要があるため，コードクローン検出よりは計算量は小さいものの，膨大な計算時間が必要である．従って，コードクローン検索ではインデックス作成など，類似度計算量の削減のための工夫が行われている．また，C を効率よく見つけるために，コードクローン検出と同様に，同値関係に基づいてコードクローン検索を近似的に行うなど様々な工夫がされている．コードクローン検出と同様に，検索結果は必ずしも同値関係を維持していない．一般的に，コードクローン検索手法で検索対象になるコードクローンは，類似性の違いによって以下の 4 種類に分類される [38]．

タイプ 1： 空白，タブ文字，改行やコメントなどを除いて一致するコードクローン．

タイプ 2： タイプ 1 の条件に加えて，リテラル，型，識別子を除いて一致するコードクローン．

タイプ 3： タイプ 2 の条件に加えて，文の変更，追加または削除など違いを含むコードクローン

タイプ 4： 同様の処理を行うが，構文的な差異を含むコードクローン

また，多くのコードクローン検索手法では，C に含まれるコードクローンを q との類似度に基づいて並べ替え，提示する．

3 コードクローン検索手法

2.1 節で説明した通り，Liu らの調査ではコード検索作業に基づいて既存のコード検索ツールを 7 つのカテゴリに分類し，コード検索ツールの研究動向及び課題，今後の研究方向性を議論した．しかし，彼らの調査はコードクローン検索分類に不正確な部分があり (問題 1)，かつコードクローン検出技術を用いた検索手法に関して詳細な解説を行っていない (問題 2)．そこで，本稿では，Liu らの調査における問題 1 と問題 2 を緩和するために，彼らの調査結果に基づいて現在まで提案されているコードクローン検索手法を紹介する．彼らはコードクローン検索に関する 9 編の文献 [3] [8] [20] [22] [23] [27] [35] [40] [51] を紹介している．本研究では，彼らがコードクローン検索ツールとして分類した文献のうち，以下の条件 1 と条件 2 を両方満たす 4 編の文献 [8] [20] [40] [51] を，コードクローン検索よりコードクローン検出作業に焦点をあてていると判断して調査対象から除外した．

条件 1：文献で，ソースコードから全てのコードクローンの対，あるいは全てのコードクローンの集合を効率良く検出するための手法が提案されている．

条件 2： 文献で提案手法をソースコードに適用し，その結果，入力として与えられたソースコードからコードクローンを正確に検出できるかを評価している．

なお，上記の 2 つの条件を満たさない，かつコードクローン検索手法を提案し，評価した 5 編の文献 [3] [22] [23] [27] [35] は調査対象に含めた．また，コードクローン検索手法の文献であり，かつ重要だと判断した文献を著者らが知る限り，調査対象文献に追加した．具体的に，まず，Liu らによりコードクローン検索に分類された文献で関連研究として挙げられている文献のうち，コードクローン検索手法を提案し，評価実験でも提案手法のコードクローン検索精度を評価している，かつ 2.1 節で説明した Liu らが調査対象文献を選定するために用いた 6 つの規準を満たす文献を，本稿の調査対象として新しく追加した．また，我々が既に知っているコードクローン検索に関する文献が，2.1 節で説明した「IC：英語で書かれていること」以外の 2 つの IC を満たし，かつ 3 つの EC に該当しない場合，その文献を調査対象として新しく追加し，調査対象文献を確定した．

3.1 コードクローン検索技術に基づいた分類

我々は，調査対象文献の中身を分析し，文献の中でクエリ q とコードクローン関係であるコード片の集合 C を検索するために主として使われているコードクローン検出技術を以下の 3 種類に分類した．

- 木構造の分析に基づく技術
- 自然言語処理に基づく技術
- 深層学習を用いた技術

2.2 節で説明したとおり，コードクローン検出手法は入力として与えられたソースコードからコードクローン関係である全てのコード片の対，あるいは全てのコード片の集合を特定する手法であった．本稿では，コードクローン検出技術はコード片の対がコードクローン関係であるか否かを判定する技術を指す．コードクローン検索手法はコードクローン検出技術により，検索対象コード片の集合 T の各要素について，クエリ q がコードクローン関係であるか否かを判定する．

表 2.2 は各手法がどのコードクローン検出技術によってコードクローンを検索しているかを表している．表 2.2 から既存のコードクローン検索手法ではコードクローン検出で用いられているグラフ表現に基づく手法が提案されていないことがわかる．その理由としてコードクローン検索では入力として構文的に不完全な 1 つのコード片も使用することが挙げられる．つまり，コードクローン検索では入力 (クエリ) として構文的に不完全な 1 つのコード片を使用するため，入力ソースコードからグラフを構築する場合，不正確なグラフが構築されるか，あるいはグラフの構築に失敗する可能性が高い．従って，コードクローン検索手

表 1 コードクローン検索手法で用いられるコードクローン検出技術

掲載年	文献	木構造の分析	自然言語処理	深層学習
2009	吉田 [53]		✓	
2010	Lee [23]	✓		
2015	Balachandran [3]	✓		
	Nishi [33]		✓	
2018	Kim [22]		✓	
	Zhou [56]		✓	✓
	Siamese [35]		✓	
2019	Fujiwara [9]		✓	✓
	Aroma [27]	✓		
2021	Bui [7]	✓		✓

法ではグラフ表現が用いられていないと考えられる．以降，各分類に属するコードクローン検索手法を紹介する．

木構造の分析に基づく技術： 木構造に基づいたコードクローン検出手法では，入力ソースコードを木構造で表現し，一致または類似した部分木をコードクローンとして検出する．例えば，Jiang らが開発した DECKARD [19] は，入力ソースコードを解析木に変換し, 解析木の部分木を特徴ベクトルで表現する．そして，特徴ベクトル間の類似度を求めることによって，コードクローンの検出を行う．木構造に基づいたコードクローン検索手法では，クエリと検索対象コード片の集合を木構造に変換し，木構造に基づいたコードクローン検出と同様の照合方法によりクエリと木構造が一致または類似しているコード片を検索対象コード片の集合から見つける．Lee らはインデックス作成技術を改善することで正確かつ高速にコードクローンを検索する手法を提案した [23]．彼らの提案手法は，インデックスの次元を削減し，$R*tree$ [5] を用いて検索対象ソースコードのインデックスを作成する．また，コードクローンを検索する際に，各データではなく，それを内包する矩形領域の先頭にのみランダムアクセスするように，データアクセス時の I/O コストを改善した．

Balachandran は抽象構文木 (Abstract Syntax Tree，以下 AST) の類似度に基づいたコードクローン検索手法を提案した [3]．検索対象の各ファイルの AST を生成し，閾値以上のサイズの AST の部分木の特徴ベクトルを生成する．クエリとして与えられたコード片を AST に変換し特徴ベクトルを生成した後に，クエリの AST の部分木を検索対象コード片のソースファイルの AST の部分木と比較する．その後，クエリの AST の部分木とマッチングするソースファイルの AST の部分木の類似度をベクトル間のユークリッド距離により近似することで，各ソースファイルの関連度スコア (Relevance Score) を算出する．最後に，マッチングされたファイルを関連度スコア順で表示する．

Luan らは解析木から抽出した特徴に基づいて構造的に類似するコードクローンを検索するツール Aroma を提案した [27]．Aroma は，検索対象の各メソッドから単純化した解析木を作成し，構文木から構造的な特徴を抽出する．また，クエリの単純化した解析木から抽出した特徴を用いてクエリの大部分を含むメソッドを検索する．その後，クエリと共通集合をもつコード片を検索結果から抽出し，抽出したコード片をクエリの類似度に基づいて並び替える．最後に，並び替えられたコード片をクラスタリングし，各クラスタ内の

全てのコード片の共通集合を取ることで類似メソッドの集合を提示する．

自然言語処理に基づく技術： 自然言語処理に基づいたコードクローン検出手法では，入力ソースコードに自然言語処理技術を適用し，入力ソースコード内の変数名や関数名などの識別子などが一定以上の割合で一致または類似しているソースコードをコードクローンとして検出する．例えば，山中らの手法[52]は，TF-IDF (Term Frequency-Inverse Document Frequency) 法[41]を使用して，ソースコード中の識別子や予約語に利用される単語に対して重み付けを行うことによって関数を特徴ベクトルに変換し，その特徴ベクトル間の類似度を求めることで関数単位のコードクローンを検出する．自然言語処理に基づいたコードクローン検索手法では，自然言語処理技術をクエリと検索対象コード片の集合に適用し，自然言語処理に基づいたコードクローン検出手法と同様の照合方法によりクエリとソースコード内の単語が一定以上の割合で一致または類似しているコード片を検索対象コード片の集合から見つける．

吉田らはコード片に含まれる識別子の類似性に基づいてコードクローンを検索する手法を提案した[53]．具体的には，まず，検索対象のソースコード集合の各関数が含む語 (識別子を分割・正規化した後の文字列) を抽出する．次に，抽出された語の集合を Jensen-Shannon divergence[24]に基づいてクラスタリングを行い，類義語を特定する．最後に，クエリとして与えられたコード片と検索対象ソースコード集合の各関数を参照することで，クエリが含む全ての語と一致もしくは類似する語を含む関数を検索し，提示する．

Nishi と Damevski は adaptive prefix filtering[49]に基づいたコードクローン検索手法及びコードクローン検出手法を提案した[33]．彼らの手法はまず，検索対象のコード片に含まれているトークン情報に基づいてデルタ転置インデックスを生成する．また，クエリとして与えられたコードブロックをトークン化し，トークンを全てのコードブロックの中の出現頻度順でソートし，先頭付近のみからトークンを取り出す．次に，取り出したトークンと一定数以上一致するトークンを含むコードブロックをコードクローンとして検索する．

Kim らは開発者向け Q&A サイト StackOverflow[†1] を用いて，タイプ 4 までのコードクローンまで検索する FaCoY を提案した[22]．FaCoY は StackOverflow の質問文と回答文に含まれているコード片を抽出して，インデックスを生成する．その後，クエリとして与えられたコード片に類似したソースコードが含まれる回答文を StackOverflow から検索する．次に，検索された回答文に対応する質問文をクエリとし，再び StackOverflow から回答文の検索を行い，検索結果の回答文に含まれるソースコードを検索結果として出力する．

Zhou らは深層学習に基づく統計的言語モデルを用いて，類似ソースコードを推薦するツール SLAMPA を開発した[56]．SLAMPA はまず，LSTM (Long-Short Term Memory recurrent neural network)[16]を用いてクエリとして与えられた不完全なコード片の意図を推測し，新しいトークン列を生成する．次に，生成されたトークン列を用いて検索対象のソースコード集合から類似ソースコードを検索して，最も類似度が高いコード片を提示する．

Ragkhitwetsagul と Krinke は，n-gram を用いて構文的に類似したメソッドを検索するツール Siamese を開発した[35]．Siamese では，検索対象であるメソッドのトークン化を行い，タイプ 3 までのコードクローンを検索するために，複数のソースコード表現を生成する．次に，クエリに対しても同様に複数のソースコード表現を生成し，生成されたソースコード表現と検索対象ソースコード集合のトークンの出現頻度を考慮しクエリから出現頻度が高いトークンを除外し，コードクローン検索を行う．最後に，Apache Lucene の TF-IDF 法に基づいたスコアを用いて，スコ

†1 https://stackoverflow.com/

アが高いメソッドの組を類似メソッドとして出力する．

深層学習を用いた技術： 深層学習に基づいたコードクローン検出手法では，ソースコードのベクトル表現を深層学習モデルに学習させ，そのモデルを用いてコードクローンを検出する．例えば，Zhao と Huang が提案した DeepSim [55] は，制御フローグラフとデータフローグラフのベクトル表現情報を基に，深層学習モデルを用いて学習することでコードクローンを検出する．深層学習に基づいたコードクローン検索手法では，深層学習に基づいたコードクローン検出手法と同様にベクトル表現の深層学習を行ったモデルを用いて，クエリと同一カテゴリと深層学習モデルにより判定されたベクトル表現のコード片を検索対象コード片の集合から見つける．藤原らは順伝播型ニューラルネットワーク (Feedforward Neural Network；以降 FFNN) [43] を用いた手法を提案した [9]．具体的には，検索対象プロジェクトから，検索クエリとして与えたコード片のコードクローンを検索する．まず，検索対象プロジェクトに含まれるコードブロックに対してミューテーションオペレータ [37] を適用することで，様々な種類の類似コードブロックを新たに作成し，それらの特徴ベクトルを学習用データセットのポジティブデータとする．このとき，類似コードブロックの集合毎にユニークなラベル l(l は自然数) を付与する．次に，検索対象プロジェクトに含まれるコードブロックのいずれともコードクローンではないコードブロックの特徴ベクトルを，学習データセットのネガティブデータとする．このとき，ネガティブデータにはラベル 0 を付与する．以上のポジティブデータとネガティブデータを説明変数，付与したラベルを目的変数として，FFNN の多クラス分類モデルの教師あり学習を行う．クエリとして与えられたコード片から作成された特徴ベクトルを学習済み FFNN モデルに与えると，入力コード片に類似している学習済コードブロックが存在する場合は，類似している学習済コードブロックに付与されたラベルを出力し，存在しない場合はラベル 0 を出力する．

Bui らは．自然言語処理の自己教師あり学習アイディアを AST に適用したソースコード表現のモデル InferCode を提案した [7]．InferCode は，AST の集合から自己教師あり学習に基づき自動的に識別された部分木を予測することで，ソースコード表現を訓練する．彼らは，大規模な Java ソースコードのエンコーダーとして TBCNN (Tree-Based Convolutional Neural Network) [32] を使用することで，InferCode モデルのインスタンスを学習し，InferCode が言語間コード検索にも有効であることを示した

3.2 コードクローン検索手法で用いられている評価指標

本節では，コードクローン検索手法がどのように評価されているかを調査する．表 2 は評価実験で使われている評価指標の大きな分類と，各々の分類に属している評価指標をまとめたものである．表 2 で，評価指標に関する大きな分類 (ランキング，分類，性能) は Liu らの研究を参考にしたものである．まず，この表で示している評価指標に関して説明する．分類に関する評価指標 *Precision* と *Recall* の算出式は以下の通りである．

$$Precision = \frac{|E \cap O|}{|O|} \tag{1}$$

$$Recall = \frac{|E \cap O|}{|E|} \tag{2}$$

式 (1) と (2) において， E は正解コード片の集合，O は提案手法によって検索されたコード片の集合とする．また，*F-measure* は Precision と Recall の調和平均によって求められる．

また，ランキングに関する指標である $Precision@k$ と $Recall@k$ は，ランキングの上位 k 件における Precision, Recall をそれぞれ意味する．また，MRR (Mean Reciprocal Rank), MAP (Mean Average Precision), $HitRate@k$ を計算する式は以下の通りである．

$$MAP = \frac{1}{Q}\sum_{i=1}^{Q} AveP(i) \tag{3}$$

表 2　評価指標

タイプ	評価指標	文献
分類	Precision	[53], [23], [3], [56], [9]
	Recall	[53], [23], [3], [22], [56], [9]
	F-measure	[53], [23], [3], [9]
ランキング	Precision@k	[22], [35]
	Recall@k	[27]
	MAP	[35]
	MRR	[7], [22], [56], [35]
	HitRate@k	[56]
性能	インデックス作成時間	[23], [33], [35]
	クエリ時間	[23], [33], [35]

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{rank_i} \tag{4}$$

$$HitRate@k = \frac{1}{Q}\sum_{i=1}^{Q}\xi(R(i), k) \tag{5}$$

式 (3) において Q は全ての検索クエリの数，$AveP(i)$ はクエリ i の平均 Precision を示す．式 (4) において $rank_i$ は検索クエリ i に対して正解コード片が検索された時の順位である．式 (5) において $R(i)$ はクエリ i の検索結果の集合，$\xi(\cdot)$ は正解コード片の順位が k 以下の場合は 1 を，そうでない場合は 0 を返す関数である．

次に，評価実験で使われている評価指標をまとめた結果を議論する．表 2 からわかるように，重複して現れている文献を除く場合，正解コード片が検索できるか (分類) に関する評価を行った文献は 6 編，正解コード片を上位に表示できるか (ランキング) に関する評価を行った文献は 5 編であり，既存研究では分類とランキングに関しては同程度評価が行われていることがわかった．また，分類に関する評価指標では，Recall は最も多く使用されていることや，ランキングに関しては MRR が最も頻繁に使用されていることがわかった．また，検索結果に関する評価だけではなくて，提案手法の性能評価として，インデックス作成にかかる時間と検索にかかる時間を評価する研究も存在していることがわかった．

4　今後の研究課題

- **多様なプログラミング言語に対応可能なコードクローン検索手法の開発：** 今まで提案されているコードクローン検索手法は主に Java で開発されたソースコードを検索対象としている．近頃，深層学習に基づいたソースコード言語のモデルを用いることで全ての言語に適用しやすい手法 [7] や，Java 以外の複数の言語を対象にコードクローンを検索する手法 [27] が提案されているが，その数は不十分である．そこで，開発者の複数の言語間のコードクローン検索や多様なプログラミング言語に対応可能 [44] なコードクローン検索手法の開発が課題として挙げられる．
- **深層学習モデルのためのデータセット構築及び改善手法の提案：** 本稿の調査により近年，深層学習を用いたコードクローン検索手法 [7] [9] が注目されていることがわかった．深層学習モデルに基づいたコード検索手法で正確なモデルを構築するためには，コードクローン検索のためのデータセットが必要である．また，構築されたデータセットを改善することが深層学習モデルの精度の向上が期待できる．従って，深層学習モデルの構築のためのデータセット構築や改善手法 [10] の提案が課題として挙げられる．

- **コードクローン検索手法の性能比較調査**：本稿で現在まで提案されているコードクローン検索手法を紹介した．しかし，これらの有効性を示すために必要なコード検索手法間の性能比較が行われていない．そこで，現在まで提案されたコードクローン検索手法の性能比較調査が研究課題として挙げられる．
- **グラフ表現に基づいたコードクローン検索手法の提案**：表 2.2 から既存のコードクローン検索手法では，コードクローン検出と異なってグラフ表現に基づいた手法が提案されていないことがわかった．その理由として 3.1 節で説明した通り，コードクローン検索では入力として構文的に不完全な 1 つのコード片を使用することが考えられる．しかし，コード検索手法で入力コード片を構文的に完全なコード片のみに想定した場合，コードクローン検出で使われているプログラム依存グラフなどを用いてコードクローンが検索できると考えられる．従って，グラフ表現に基づいたコードクローン検索手法の提案が課題として挙げられる．
- **検索結果コード片の有効性に関する評価の実施**：表 2 から既存研究ではランキング，分類，検索時間に属する評価指標が広く用いられていることがわかった．しかし，提案手法によって品質の低いコード片が検索されそのコード片が開発タスクに用いられた場合，ソフトウェアの品質や生産性の低下が起きる可能性が高い．従って，提案手法によって有効なコードクローンが検索されたかを評価するために検索結果のコード片の再利用性メトリック [31] や可読性メトリクス [42] などを用いて検索結果の有効性に関する評価を実施する必要がある．そのため，検索結果の有効性に関する評価実験が課題として挙げられる．

5 関連研究

本章では，現在まで国際会議及びジャーナルで出版されているコードクローンに関するサーベイ論文を関連研究として紹介する．コードクローンの管理作業として，クローンリファクタリング (コードクローン集合の中に含まれるコードクローンを 1 つのメソッドなどに集約すること) やコードクローン追跡などがあり，これらの作業を支援するために多くの技術が提案されている．Mondal らは，クローンリファクタリングと追跡に関する既存文献を調査し，クローンリファクタリングと追跡ツールのための品質評価特徴に基づいて既存ツールを比較した [30]．また，検出されたコードクローンを可視化することでコードクローン管理を支援することができるため，様々なコードクローン可視化手法も提案されている．Basit らは 2015 年にコードクローン可視化文献に関するサーベイを実施し，コードクローンの可視化タイプと可視化が提供する情報について議論した [4]．Hammad らは 2020 年にコードクローン可視化手法のシステマティックマッピングを行った [14]．彼らは，可視化要素と情報に従って，既存のコードクローン可視化手法を分類し，それぞれの分類固有の長所と短所を議論した．堀田らは 2014 年にコードクローン管理支援技術文献のサーベイを行った [17]．彼らはコードクローン管理支援技術のうち，生成，拡張の抑止，分析の効率化，不具合の混入防止，及び検出分類に属する文献を紹介している．Bharti らは 2020 年に総合開発環境でプラグインとして開発されたリアルタイムコードクローン管理ツールの文献に関する体系的レビューを行った [6]．コードクローンの進化に関する調査も活発に行われている．Pate らは 2013 年にコードクローンの進化に関する文献の体系的レビューを行った [34]．また，彼らは体系的レビュー結果に基づいて既存文献で使用されているクローン進化を調査するために使用された手法，既存研究で明らかになったコードクローンの進化パタン，一貫していない変化が行われたコードクローンの進化の調査結果を紹介した．

2007 年に Roy と Cordy はコードクローン検出技術とツールをサーベイした [39]†2．彼らは，コードクローンと関係のある用語と定義をまとめて，既存のコードクローン検出技術とツールをコードクローン検出単位，検出結果の評価方法や応用及び可視化技術などによって分類した．肥後らは 2008 年にコー

†2 この文献はテクニカルレポートだが，有名なコードクローンサーベイ論文であるため紹介する．

ドクローンの検出とその関連技術の研究成果を紹介した[15]．具体的に，既存のコードクローン検出技術を検出単位に基づいて分類し，検出手法の比較とその結果，可視化手法及び集約，修正支援などを紹介した．Roy らは 2009 年にコードローン検出技術とツールに関する包括的な調査を行った[38]．彼らは，コードクローンの概念、一般的な検出プロセス、現在の技術やツールの検出単位による分類を紹介し，既存の技術やツールを様々な要因 (例：使用方法，言語，利用シナリオ) に基づいて分類した．それ以降，Rattan らは 2013 年にクローン検出文献に関する体系的レビューを行った[36]．また，2016 年に Gautam と Saini[11]，Sheneamer と Kalita[45] が，2018 年に Gupta と Suri[13] が，2019 年に Wang ら[50]，Ain ら[1]，Min と Li Ping[28] が，コードクローン検出手法に関するサーベイを行った．これらの文献は，既存のコードクローン検出手法を検出単位によって分類し，紹介した．なお，特定の技術に基づいたコードクローン検出手法に関する文献のみを調査したサーベイもある．Kaur らは機械学習に基づくコードクローン検出手法を紹介している[21]．Zhang と Sakurai は脆弱性を含むコードの解析に基づいたコードクローン検出技術を紹介した[54]．このように現在までコードクローン管理手法や可視化手法，進化調査および検出手法に関する様々なサーベイが行われている．本稿は既存のコードクローンに関するサーベイと異なり，コードクローン検索に重点を置いてコードクローン検索の既存文献に関するサーベイを行っている．

6 まとめ

本稿では，コード検索手法の中でも，コードクローン検索手法を中心に紹介を行った．まず，Liu らが実施したコード検索ツールに関する系統的レビューを紹介した．その後，コードクローン検索手法の中で使用されているコードクローン検索技術を木構造の分析に基づいた技術，自然言語処理に基づいた技術，深層学習を用いた技術に分類し，それぞれの分類に属するコードクローン検索手法について紹介した．その後，コードクローン検索研究で使用されている評価指標を分析し，既存研究では分類とランキングに関しては同程度評価が行われていることがわかった．最後に，今後の研究の方向性について考察を行った．

コードクローン検索手法に関する研究は現在早期段階であり今後より多くの手法の提案が期待される．本論文が，コードクローン検索手法の発展の一助となれば幸いである．

謝辞 本研究は JSPS 科研費 JP18H04094，JP19K20240，JP20K11745，21H03416 の助成を受けた．

参考文献

[1] Ain, Q. U., Butt, W. H., Anwar, M. W., Azam, F., and Maqbool, B.: A Systematic Review on Code Clone Detection, *IEEE Access*, Vol. 7 (2019), pp. 86121–86144.

[2] Bajracharya, S. K., Ossher, J., and Lopes, C. V.: Leveraging Usage Similarity for Effective Retrieval of Examples in Code Repositories, in *18th International Symposium on Foundations of Software Engineering*, FSE '10, 2010, pp. 157–166.

[3] Balachandran, V.: Query by example in large-scale code repositories, in *4th International Conference on Software Maintenance and Evolution*, ICSME '15, 2015, pp. 467–476.

[4] Basit, H. A., Hammad, M., and Koschke, R.: A survey on goal-oriented visualization of clone data, in *3rd Working Conference on Software Visualization*, VISSOFT '15, 2015, pp. 46–55.

[5] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B.: The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles, in *ACM SIGMOD International Conference on Management of Data*, SIGMOD '90, 1990, pp. 322–331.

[6] Bharti, S. and Singh, H.: Proactively managing clones inside an IDE: a systematic literature review, *Int. J. Comput. Appl.*, (2020), pp. 1–20.

[7] Bui, N. D. Q., Yu, Y., and Jiang, L.: InferCode: Self-Supervised Learning of Code Representations by Predicting Subtrees, in *43rd International Conference on Software Engineering (accepted for publication)*, ICSE '21, 2021.

[8] Chen, L., Ye, W., and Zhang, S.: Capturing Source Code Semantics via Tree-Based Convolution over API-Enhanced AST, in *16th ACM International Conference on Computing Frontiers*, CF '19, 2019, pp. 174–182.

[9] Fujiwara, Y., Yoshida, N., Choi, E., and Inoue, K.: Code-to-Code Search Based on Deep Neural Network and Code Mutation, in *13th International Workshop on Software Clones*, IWSC '19, 2019, pp. 1–7.

[10] 藤原裕士, 崔恩瀞, 吉田則裕, 井上克郎: 深層学習

を用いたソースコード分類のための動的な学習用データセット改善手法の提案, 電子情報通信学会論文誌 *D*, Vol. J104-D, No. 4 (2021), pp. 275–284.

[11] Gautam, P. and Saini, H.: Various Code Clone Detection Techniques and Tools: A Comprehensive Survey, *Smart Trends in Information Technology and Computer Communications*, 2016, pp. 655–667.

[12] Gu, X., Zhang, H., and Kim, S.: Deep Code Search, in *40th International Conference on Software Engineering*, ICSE '18, 2018, pp. 933–944.

[13] Gupta, A. and Suri, B.: A Survey on Code Clone, Its Behavior and Applications, *Networking Communication and Data Knowledge Engineering*, 2018, pp. 27–39.

[14] Hammad, M., Basit, H. A., Jarzabek, S., and Koschke, R.: A systematic mapping study of clone visualization, *Comput. Sci. Rev.*, Vol. 37 (2020), pp. 1–55.

[15] 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, 電子情報通信学会論文誌. *D,* 情報・システム, Vol. 91, No. 6 (2008), pp. 1465–1481.

[16] Hochreiter, S. and Schmidhuber, J.: Long Short-Term Memory, *Neural Comput.*, Vol. 9, No. 8 (1997), pp. 1735–1780.

[17] 堀田圭佑, 肥後芳樹, 楠本真二: 生成抑止，分析効率化，不具合検出を中心としたコードクローン管理支援技術に関する研究動向, コンピュータ ソフトウェア, Vol. 31, No. 1 (2014), pp. 14–29.

[18] 井上克郎, 神谷年洋, 楠本真二: コードクローン検出法, コンピュータソフトウェア, Vol. 18, No. 5 (2001), pp. 47–54.

[19] Jiang, L., Misherghi, G., Su, Z., and Glondu, S.: DECKARD: Scalable and Accurate Tree-Based Detection of Code Clones, in *29th International Conference on Software Engineering*, ICSE '07, 2007, pp. 96–105.

[20] Kamiya, T., Kusumoto, S., and Inoue, K.: CCFinder: a multilinguistic token-based code clone detection system for large scale source code, *IEEE Trans. Softw. Eng.*, Vol. 28, No. 7 (2002), pp. 654–670.

[21] Kaur, A., Sharma, S., and Saini, M.: Code Clone Detection Using Machine Learning Techniques: A Systematic Literature Review, *Int. J. Open Source Softw. Process.*, Vol. 11, No. 2 (2020), pp. 49–75.

[22] Kim, K., Kim, D., Bissyandé, T. F., Choi, E., Li, L., Klein, J., and Traon, Y. L.: FaCoY: A Code-to-code Search Engine, in *40th International Conference on Software Engineering*, ICSE '18, 2018.

[23] Lee, M.-W., Roh, J.-W., Hwang, S.-w., and Kim, S.: Instant Code Clone Search, in *18th International Symposium on Foundations of Software Engineering*, FSE '10, 2010, pp. 167–176.

[24] Lin, J.: Divergence Measures Based on the Shannon Entropy, *IEEE Trans. Inf. Theor.*, Vol. 37, No. 1 (2006), pp. 145–151.

[25] Linstead, E., Bajracharya, S., Ngo, T., Rigor, P., Lopes, C., and Baldi, P.: Sourcerer: Mining and Searching Internet-Scale Software Repositories, *Data Min. Knowl. Discov.*, Vol. 18, No. 2 (2009), pp. 300–336.

[26] Liu, C., Xia, X., Lo, D., Gao, C., Yang, X., and Grundy, J.: Opportunities and Challenges in Code Search Tools, 2020. arXiv:2011.02297.

[27] Luan, S., Yang, D., Barnaby, C., Sen, K., and Chandra, S.: Aroma: Code Recommendation via Structural Code Search, in *Proc. ACM Program. Lang.*, Vol. 3, No. OOPSLA(2019).

[28] Min, H. and Li Ping, Z.: Survey on Software Clone Detection Research, in *3rd International Conference on Management Engineering, Software Engineering and Service Sciences*, ICMSS '19, 2019, pp. 9–16.

[29] Mohagheghi, P., Conradi, R., Killi, O. M., and Schwarz, H.: An Empirical Study of Software Reuse vs. Defect-Density and Stability, in *26th International Conference on Software Engineering*, ICSE '04, 2004, pp. 282–292.

[30] Mondal, M., Roy, C. K., and Schneider, K. A.: A survey on clone refactoring and tracking, *J. Syst. Softw.*, Vol. 159 (2020).

[31] Moreno, L., Bavota, G., Di Penta, M., Oliveto, R., and Marcus, A.: How Can I Use This Method?, in *37th International Conference on Software Engineering - Volume 1*, ICSE '15, 2015, pp. 880–890.

[32] Mou, L., Li, G., Zhang, L., Wang, T., and Jin, Z.: Convolutional Neural Networks over Tree Structures for Programming Language Processing, in *30th AAAI Conference on Artificial Intelligence*, AAAI '16, 2016, pp. 1287–1293.

[33] Nishi, M. A. and Damevski, K.: Scalable code clone detection and search based on adaptive prefix filtering, *J. Syst. Softw.*, Vol. 137 (2018), pp. 130–142.

[34] Pate, J. R., Tairas, R., and Kraft, N. A.: Clone evolution: a systematic review, *J. Softw.: Evol. Process*, Vol. 25, No. 3 (2013), pp. 261–283.

[35] Ragkhitwetsagul, C. and Krinke, J.: Siamese: scalable and incremental code clone search via multiple code representations, *Empir. Softw. Eng. J.*, (2019), pp. 2236–2284.

[36] Rattan, D., Bhatia, R., and Singh, M.: Software clone detection: A systematic review, *Inf. Softw. Technol.*, Vol. 55, No. 7 (2013), pp. 1165–1199.

[37] Roy, C. K. and Cordy, J. R.: A Mutation/Injection-Based Automatic Framework for Evaluating Code Clone Detection Tools, in *International Conference on Software Testing, Verification, and Validation Workshops*, ICSTW '09, 2009, pp. 157–166.

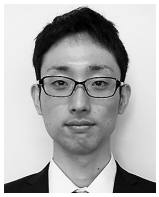
[38] Roy, C. K., Cordy, J. R., and Koschke, R.: Comparison and evaluation of code clone detection techniques and tools: A qualitative approach, *Sci. Comput. Program.*, Vol. 74, No. 7 (2009), pp. 470–495.

[39] Roy, C. K. and Cordy, J. R.: A Survey on Software Clone Detection Research, *School of Computing TR 2007-541, Queen's University*, Vol. 115

(2007).
[40] Sajnani, H., Saini, V., Svajlenko, J., Roy, C. K., and Lopes, C. V.: SourcererCC: Scaling Code Clone Detection to Big-Code, in *38th International Conference on Software Engineering*, ICSE '16, 2016, pp. 1157–1168.
[41] Salton, G. and McGill, M. J.: *Introduction to Modern Information Retrieval*, McGraw-Hill, Inc., 1986.
[42] Scalabrino, S., Linares-Vásquez, M., Poshyvanyk, D., and Oliveto, R.: Improving code readability models with textual features, in *24th International Conference on Program Comprehension*, ICPC '16, 2016, pp. 1–10.
[43] Schmidhuber, J.: Deep learning in neural networks: An overview, *Neural Netw.*, Vol. 61 (2015), pp. 85–117.
[44] 瀬村雄一, 吉田則裕, 崔恩瀞, 井上克郎: 多様なプログラミング言語に対応可能なコードクローン検出ツール CCFinderSW, 電子情報通信学会論文誌 *D*, Vol. J103-D, No. 4 (2020), pp. 215–227.
[45] Sheneamer, A. and Kalita, J.: A Survey of Software Clone Detection Techniques, *International Journal of Computer Applications*, Vol. 137 (2016), pp. 1–21.
[46] Stolee, K. T., Elbaum, S., and Dobos, D.: Solving the Search for Source Code, *ACM Trans. Softw. Eng. Methodol.*, Vol. 23, No. 3 (2014).
[47] Stolee, K. T., Elbaum, S., and Dwyer, M. B.: Code Search with Input/Output Queries, *J. Syst. Softw.*, Vol. 116, No. C (2016), pp. 35–48.
[48] Thummalapenta, S. and Xie, T.: Parseweb: A Programmer Assistant for Reusing Open Source Code on the Web, in *22nd International Conference on Automated Software Engineering*, ASE '07, 2007, pp. 204–213.
[49] Wang, J., Li, G., and Feng, J.: Can We Beat the Prefix Filtering? An Adaptive Framework for Similarity Join and Search, in *ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, 2012, pp. 85–96.
[50] Wang, Y., Liu, D., and Hou, M.: Study of Clone Code Detection Method, in *3rd International Conference on Computer Engineering, Information Science and Application Technology*, ICCIA '19, 2019, pp. 414–423.
[51] White, M., Tufano, M., Vendome, C., and Poshyvanyk, D.: Deep learning code fragments for code clone detection, in *31st IEEE/ACM International Conference on Automated Software Engineering*, ASE '16, 2016.
[52] 山中裕樹, 崔恩瀞, 吉田則裕, 井上克郎: 情報検索技術に基づく高速な関数クローン検出, 情報処理学会論文誌, Vol. 55, No. 10 (2014), pp. 2245–2255.
[53] 吉田則裕, 服部剛之, 早瀬康裕: 類義語の特定に基づく類似コード片検索法, 情報処理学会論文誌, Vol. 50, No. 5 (2009), pp. 1506–1519.
[54] Zhang, H. and Sakurai, K.: A Survey of Software Clone Detection From Security Perspective, *IEEE Access*, Vol. 9 (2021), pp. 48157–48173.
[55] Zhao, G. and Huang, J.: DeepSim: Deep Learning Code Functional Similarity, in *26th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE '18, 2018, pp. 141–151.
[56] Zhou, S., Zhong, H., and Shen, B.: SLAMPA: Recommending Code Snippets with Statistical Language Model, in *25th Asia-Pacific Software Engineering Conference*, 2018, pp. 79–88.

崔　恩瀞

2015 年大阪大学大学院情報科学研究科博士後期課程修了．同年同大学大学院国際公共政策研究科助教．2016 年奈良先端科学技術大学院大学情報科学研究科助教．2018 年より同大学先端科学技術研究科助教 (改組による)．2018 年より京都工芸繊維大学情報工学・人間科学系助教．博士 (情報科学)．コードクローン管理やリファクタリング支援手法に関する研究に従事．



藤原裕士

2017 年大阪大学基礎工学部情報科学科卒業，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修了．現在は大阪大学大学院情報科学研究科博士後期課程に在学中．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．

吉田則裕

2009 年大阪大学大学院情報科学研究科博士後期課程修了．同年日本学術振興会特別研究員 (PD)．2010 年奈良先端科学技術大学院大学情報科学研究科助教．2014 年名古屋大学大学院情報科学研究科附属組込みシステム研究センター准教授．2017 年より同大学大学院情報学研究科附属組込みシステム研究センター准教授 (改組による)．博士 (情報科学)．コードクローン分析手法やリファクタリング支援手法に関する研究に従事．

水野 修

1999 年大阪大学大学院基礎工学研究科助手．2001 年博士 (工学) 大阪大学．2010 年京都工芸繊維大学大学院工芸科学研究科准教授．2017 年同情報工学・人間科学系 教授．主にソフトウェアのバグ検出手法，ソフトウェアリポジトリのマイニングに関する研究に従事．電子情報通信学会，情報処理学会，IEEE 各会員．