修士学位論文

題目

大規模言語モデルの出力を介したコード間の機能的類似性の定量化

指導教員

楠本 真二 教授

報告者

吉岡 遼

令和 6 年 2 月 1 日

大阪大学 大学院情報科学研究科

コンピュータサイエンス専攻

令和 5 年度 修士学位論文

大規模言語モデルの出力を介したコード間の機能的類似性の定量化

吉岡 遼

内容梗概

ソフトウェア工学において，ある 2 つのコードがどの程度類似しているかの定量化は重要である．コード間の類似性の定量化により，コードクローン検出やコードサーチ，リファクタリングでの活用が期待されるためである．コード間の類似性には，構文的類似性と機能的類似性の 2 種類がある．構文的類似性とは，コードの構文や構造等がどの程度類似しているのかを表す．それに対して，機能的類似性とは，コードの持つ機能がどの程度類似しているのかを表す．本研究では，機能的類似性を研究対象とする．これまでにコード間の機能的類似性の計測を可能とする手法はいくつか提案されている．しかし，既存の手法では機能的類似性を正確に計測できない場合がある．筆者は，この原因が構文的類似性が機能的類似性に影響を与えているためだと考えた．そこで，本研究では構文的類似性が与える影響を抑制し機能的類似性を定量化する手法を提案する．提案手法では，大規模言語モデルによってコードの機能を説明させ，説明文の意味的類似性に基づき機能的類似度を計測する．提案手法では，機能的類似性の算出にコードのトークンの並びや抽象構文木といった構文的類似性に強く関連する要素を用いないため，構文的類似性が機能的類似性に与える影響を軽減できると考えた．本研究では提案手法をクローン検出ツールとして評価し，既存手法である InferCode と ASTNN よりも高い精度でクローンを検出できると確認した．また，得られた結果を分析し，提案手法において構文的類似性が機能的類似性に与える影響を抑えられていると確認した．

主な用語

機能的類似性，大規模言語モデル，コードクローン，コードサーチ

目次

図目次

表目次

1　はじめに

ソフトウェア工学において，ある 2 つのコードがどの程度類似しているかの定量化は重要である [1]．コード間の類似性の定量化により，コードクローン検出 [2,3] やコードサーチ [4,5,6] での活用が期待されるためである．コードクローンとは，ソフトウェアのソースコード中に存在する類似したコード片である．コードクローンはソフトウェアの開発や保守に悪影響を与えると報告されている [3,7]．コード間の類似性の定量化により，自動的なコードクローンの検出が可能となる．また，コードサーチとは入力されたクエリに関連したコードを検索し出力する技術である [6]．クエリにはコードそのものを与える場合がある．このような場合に，コード間の類似性の定量化によって与えられたコードと類似性の高いコードを出力できる．

コード間の類似性には構文的類似性と機能的類似性がある [2,8]．構文的類似性とはコードの構文や構造等がどの程度類似しているかを表す．それに対して，機能的類似性とはコードの持つ機能がどの程度類似しているかを表す．既存の手法の多く [3,9,10,11,12,13,14,15] はコード間の構文的類似性に着目しており，コード間の機能的類似性の計測は依然として困難な課題であると報告されている [1]．そのため，本研究では機能的類似性の計測を目的とする．

これまでにコード間の機能的類似性の計測を可能とする手法はいくつか提案されている．例えば，Bui らにより，InferCode という手法が提案されている [16]．InferCode は木構造に基づく畳み込みニューラルネットワークと抽象構文木を利用した事前学習済モデルであり，抽象構文木を活用してコード表現を学習する．InferCode はコードの埋め込み表現を導出でき，それらのコサイン類似度の算出によってコード間の機能的類似性を計測できる．他にも，Zhang らにより，ASTNN という手法が提案されている [17]．ASTNN とは，回帰ニューラルネットワークと抽象構文木を活用したモデルであり，あるコードのペアがどの程度類似しているか算出できる．しかし，InferCode や ASTNN では，機能的類似性を正確に計測できない場合がある．これまでの研究により，InferCode や ASTNN は機能的に等価でないコードを機能的に等価であると誤って検出してしまうため，機能的に等価なコードを適切に検出するためには新しい手法の開発が必要であると報告されている [18]．この原因は，InferCode や ASTNN が抽象構文木を利用しているためだと考えた．抽象構文木は構文的類似性に強く関連するため，機能的類似性と構文的類似性をうまく切り離せず，機能的類似性を十分正確に計測できないと考えた．

そこで，本研究では構文的類似性が与える影響を抑制し機能的類似性を定量化する手法を提案する．提案手法では大規模言語モデルである GPT-3.5 と，自然言語モデルの Sentence-BERT [19] を活用し，コード間の機能的類似性を定量化する．GPT-3.5 とは，OpenAI 社により公開された大規模言語モデルであり，自然言語処理におけるタスク以外にも，コードクローン検出やソースコードの要約 [20] と

いったソフトウェア工学における活用が可能である．また，Sentence-BERT とは自然言語処理モデルであり，入力として与えた文の意味に基づく埋め込み表現を導出できる．提案手法では，GPT-3.5 によりコードの機能を説明し，出力された説明文の意味的な類似度を Sentence-BERT を活用し計測することでコード間の機能的類似度を算出する．

提案手法のキーアイデアは，GPT-3.5 によるコードの説明文を介したコード間の機能的類似性の計測である．提案手法ではコードのトークンの並びや抽象構文木といった構文的類似性と強く関連する要素を直接的に機能的類似性の導出に利用しない．そのため，構文的類似性と機能的類似性を切り離し，正確に機能的類似性を計測できると考えた．

本研究では提案手法をクローン検出ツールとして評価した．その結果，提案手法は既存手法である InferCode や ASTNN よりも高い精度でクローンを検出できると確認した．得られた結果を分析し，提案手法は InferCode や ASTNN と比べ，構文的類似性が機能的類似性に与える影響を抑えられていると確認した．

以降，2 節では提案手法や提案手法の評価と深く関連する技術ついて説明する．3 節では提案手法について述べ，4 節では本研究における Research Questions を紹介する．5 節では本研究で行った実験について述べ，6 節では得られた実験結果に対する考察を記す．7 節では本研究における妥当性の脅威を述べ，8 節では本研究のまとめと今後の課題を述べる．

2 準備

ここでは，提案手法の評価方法と深く関連するコードクローン検出と，提案手法で用いる GPT-3.5 と Sentence-BERT，Sentence-BERT の元となった BERT について説明する．

2.1 コードクローン検出

コードクローン（以降，クローン）とは，ソフトウェアのソースコード中に存在する類似，あるいは一致したコード片である．クローンはソフトウェアの開発や保守に悪影響を与えるとこれまでの研究により報告されている [3, 7]．クローンは下記の 4 種類に分類される [2, 21]．

Type-1： 空白やコメント，改行を除き，完全に一致するクローン．

Type-2： 識別子名，空白やコメント，改行を除き，一致するクローン．

Type-3： プログラム文の変更，追加，削除が行われた，識別子名，空白，コメント，改行を除き一致するクローン．

Type-4： 同じ機能を有しているが，構文的に異なるクローン．

`Type-1`，`Type-2`，`Type-3` のクローンが構文的に類似したクローンであり，`Type-4` が機能的に類似したクローンである．機能的に類似したクローンは，コードの振る舞いが一致しているに過ぎず，トークン列や命令文などの構文的構造が異なるため，検出難度が構文的に類似したクローンよりも非常に高くなる [21]．本研究における提案手法はコード間の機能的類似性を定量化するため `Type-4` クローンの検出に活用できる．

2.2 GPT-3.5

GPT-3.5 とは，OpenAI 社によって作成された GPT-3 モデルを元にした大規模言語モデルである．GPT-3.5 を利用した ChatGPT は，与えられたプロンプトに対して人間と同等の回答を生成可能である [22]．GPT-3.5 には `gpt-3.5-turbo` や `gpt-3.5-turbo-1106` といったいくつかのモデルが存在する．これらのモデルのうち，`gpt-3.5-turbo` の能力が最も高いと報告されている [23]．

2.3 BERT

BERT（Bidirectional Encoder Representations from Transformers）とは，Transformer を活用した自然言語モデルであり，双方向性を持つことを特徴とする [24]．BERT 以前の多くの言語モデルは，事前学習に単方向性のタスクを利用しており，文脈を前から後ろ，あるいは後ろから前にしか考慮できなかったのに対し，BERT は事前学習に双方向性のタスクを利用しているため前後の文脈を考慮でき

る．自然言語処理に関する実験において広く利用され，数多くの研究が BERT の分析や改良の提案をしていると報告されている [25].

2.4 Sentence-BERT

Sentence-BERT [19] は，BERT を Siamese-network [26] を活用しファインチューニングした自然言語処理モデルである．入力された文の意味に基づく埋め込み表現を導出できる．得られた埋め込み表現のコサイン類似度やマンハッタン距離，ユークリッド距離の計算によって元の文の意味的な類似度を算出できる．

前述した BERT は，2 つの文を入力として受け取りそれらの類似度を算出するため，膨大な数の文の意味的な類似度を比較するには適切でない．文の組み合わせが膨大な場合に全ての組み合わせでの類似度を算出すると，膨大な計算量が必要となる．

Sentence-BERT は，文を入力して得られた埋め込み表現を算出し，それらのコサイン類似度等の計測により文の類似度を計算できる．そのため，類似度の計算に要する計算量を抑えられる．例えば，10,000 個の文にて，最も類似度が高い文のペアを取り出すというタスクでは，BERT は ${}_{10,000}C_2 = 10,000 * (10,000 - 1)/2 = 49,995,000$ 回の類似度の算出（約 65 時間）が必要であるのに対し，Sentence-BERT では 10,000 回の埋め込み表現の算出（約 5 秒）とコサイン類似度の算出（約 0.01 秒）で済むと報告されている [19].

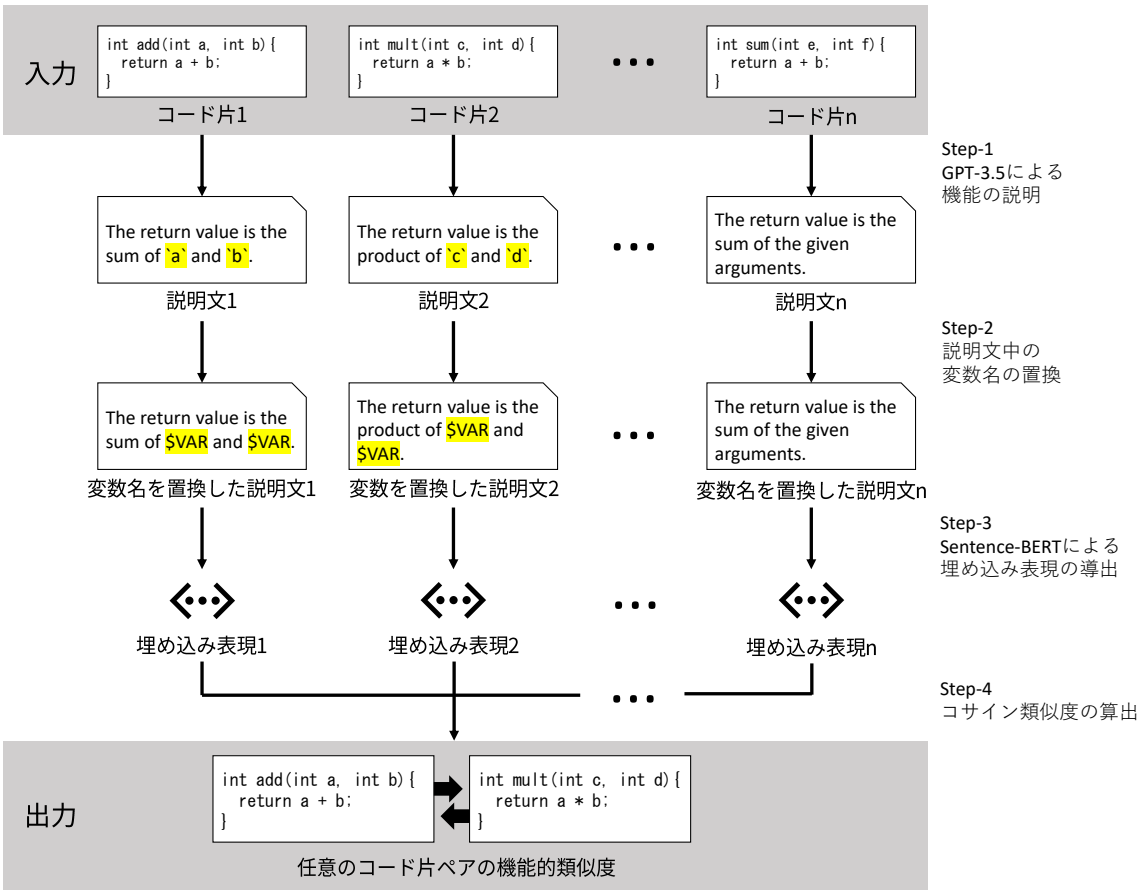


図 1: 提案手法の流れ

3 提案手法

図 1 に提案手法の流れを示す．提案手法は 4 つのステップからなる．それぞれのステップについて述べる．

Step-1： GPT-3.5 にコードの機能を説明させる．

Step-2： 出力された説明文に含まれる変数名を`$VAR` で置換し正規化する．

Step-3： Sentence-BERT を用いて，Step-2 にて得られた説明文の埋め込み表現を算出する．

Step-4： Step-3 で得られた埋め込み表現のコサイン類似度を求める．

Step-1 にて GPT-3.5 に与えるプロンプトは，対象となるコードの特徴や提案手法の活用方法により異なる．与えるプロンプトの詳細は，本研究における実験内容を記した 5 節に記す．Step-2 にて変数名を正規化する理由は機能的類似度を算出する上で変数名がノイズになると考えたためである．変数名は

コードの機能を把握するためには重要な要素であり，GPT-3.5 がコードの機能を正確に説明する上で有用な要素だと考えた．しかし，変数名はコードの機能そのものには影響を与えないため，出力された説明文に含まれる変数名は，コード間の機能的類似度を算出する上でノイズとなる可能性がある．そのため，GPT-3.5 によってコードの機能を説明させた後にコードの変数名を正規化した．図 1 の Step-1 の出力である説明文 1，2 の黄色でハイライトした部分が変数名である．この部分を，`$VAR` で置換し正規化する．Step-4 にて求まったコサイン類似度が提案手法の出力であり，コード間の機能的類似度を表す．

提案手法のキーアイデアは，コードの説明文を介した機能的類似性の定量化である．提案手法では，コードのトークンの並びや抽象構文木といった，コード間の構文的類似性に深く関連する要素を機能的類似性の算出に利用しない．そのため，構文的類似性が機能的類似性に与える影響を抑えられ，より正確に機能的類似性を計測できると考えた．

提案手法は，入力として与えられるコードが GPT-3.5 によって説明できる限り，コンパイル不可能なコードや抽象構文木を作成できないようなコード，作成途中のコードであっても利用できる．ASTNN や InferCode では抽象構文木を作成する必要があるため，提案手法の方が入力とするコードに対する制約が少なく，より多くのコードを対象として機能的類似性を計測できる．

4 Research Questions

本研究では，提案手法をクローン検出ツールとして評価する．高い精度でクローンを検出できる場合，提案手法の定める機能的類似度は妥当といえる．提案手法をクローン検出ツールとして評価するため，以下の Research Questions を定める．

RQ1： InferCode，ASTNN と比べた検出精度の調査

提案手法と InferCode，ASTNN をクローン検出ツールとして活用し，検出精度を比較する．提案手法と InferCode，ASTNN はコード間の機能的類似度を算出可能である．算出された類似度が，設けた閾値を超えている場合にはクローン，超えていない場合にはクローンでないと判断する．InferCode や ASTNN よりも提案手法の検出精度が優れていた場合，提案手法が定めるコード間の機能的類似度はこれらの手法と比べて妥当であるといえる．

RQ2： GPT-3.5 に直接類似度を算出させた場合と比べた検出精度と実行時間の調査

提案手法による機能的類似度と，GPT-3.5 のプロンプトから直接コード間の機能的類似度を出力させた場合の機能的類似度をクローン検出精度と実行時間の観点から評価する．検出精度以外に，実行時間を比較する理由を述べる．提案手法では，GPT-3.5 により出力させたコードの説明文の Sentence-BERT による埋め込み表現を記録し保持できる．一度コードの埋め込み表現を記録すれば，後は得られた埋め込み表現のコサイン類似度を計算するだけでコード間の機能的類似度を算出できる．一方で，GPT-3.5 のプロンプトから直接コード間の機能的類似度を算出させる場合，算出したいペア毎に GPT-3.5 にリクエストを送信する必要がある．例えば，10,000 個のコードにて，すべてのペアの類似度を計測するというタスクでは，GPT-3.5 へリクエストを送る回数が提案手法では 10,000 回で済むが，GPT-3.5 へ直接機能的類似度を算出させる場合には ${}_{10,000}C_2 = 10,000 * (10,000 - 1)/2 = 49,995,000$ 回と，非常に多くのリクエストを送信する必要がある．よって，提案手法と GPT-3.5 に直接機能的類似度を算出させる場合で，実行時間に大きな差が生じると考えた．したがって，本調査では，検出精度のみでなく実行時間も比較する．

RQ3： Chain-of-Thought により検出精度は向上するかの調査

GPT-3.5 へ与えるプロンプトに Chain-of-Thought を導入し，クローン検出の精度が向上するか調査する．Chain-of-Thought とは，大規模言語モデルにおいて，最終的な答えを導くための中間推論を段階的に重ね，回答を導く手法である [27]．これまでの研究により，Chain-of-Thought は複雑な推論ベンチマークにおいて，大幅な性能向上をもたらすことが報告されている [28, 29, 30]．提案手法において

も，同様に Chain-of-Thought がクローン検出精度を高めるか調査するため本 RQ を設定した.

5　実験

提案手法をクローン検出ツールとして活用し，設定した RQ に回答するための実験を行った．実験で得られた結果を本節に記す．

5.1　実験対象

実験対象には FEMPDataset [18] を利用する．FEMPDataset とは，機能的に等価なメソッドペアを収集したデータセットである．機能的に等価なメソッドペアとは，同じ引数に対して同じ返り値を返すようなメソッドペアのことである．FEMPDataset は 4 つのステップにより構築される．それぞれのステップについて説明する．

Step-1：　IJaDataset [31] に含まれるメソッドを戻り値の型と引数の型に基づいて分類する．

Step-2：　各メソッドからテストケースを生成する．

Step-3：　生成したテストをメソッド間で相互に実行し，同じ機能を持つメソッドペアの候補を取得する．すなわち，メソッド A から生成したテスト A と，メソッド B から生成したテスト B があった場合に，メソッド A に対してテスト B を，メソッド B に対してテスト A を実行し，それぞれのテストが成功した場合に，メソッド A とメソッド B を同じ機能を持つメソッドペアの候補として収集する．

Step-4：　同じ機能を持つメソッドペアの候補を目視で確認し，真に同じ機能を持つか判定する．

FEMPDataset には，目視確認の結果機能的に等価だったメソッドペアだけでなく，目視確認の結果機能的に等価でなかったメソッドペアも含まれる．また，テストの相互実行に失敗したメソッドペアは FEMPDataset に直接的には含まれていないが，FEMPDataset のデータからテストの相互実行に失敗したメソッドペアを導出できる．本研究では，FEMPDataset に含まれるメソッドペアを以下のように呼称する．

等価ペア：　テストの相互実行に通過し，かつ目視確認の結果機能等価と判断されたペア．

類似ペア：　テストの相互実行に通過したが，目視確認の結果機能等価でないと判断されたペア．

不等価ペア：　テストの相互実行に失敗したペア．

これらの等価ペア，類似ペア，不等価ペアはすべて `void` 型以外の型の返り値を持つという特徴がある．本研究では等価ペア 1,300 ペア，類似ペア 700 ペア，不等価ペア 700 ペアの合計 2,700 ペアを実験対象とする．これまでの研究 [18] を参考に，等価ペアをクローンとして検出するべきペア，類似ペアと不等価ペアをクローンとして検出すべきでないペアとした．

```
# 質問例
Explain the return value of the following program in one sentence.
Do not include any information other than the return value, such as the process flow, in your answer.
‘‘‘
String getArg(String[] args,String arg,String def){
  for (int i=0; i < args.length; i++) if (args[i].equals(arg)) return args[i + 1];
  return def;
}
‘‘‘
# 回答例
The return value of the program is the value of the argument immediately following the specified 'arg' in the 'args' array, or 'def' if 'arg' is not found in 'args'.

# 質問
Explain the return value of the following program in one sentence.
Do not include any information other than the return value, such as the process flow, in your answer.

% 機能を説明させたいメソッドを与える
```

図 2: GPT-3.5 へ与えるプロンプト

5.2 評価指標

これまでの研究 [18] を参考に，recall_E，recall_I，accuracy を利用する．それぞれの評価指標を以下にまとめる．

recall_E： クローンであると判断したペアに占める真にクローンだったペアの割合

recall_I： クローンでないと判断したペアに占める真にクローンでなかったペアの割合

accuracy： クローンである/クローンでないと正しく判断できたペアの割合

5.3 利用する GPT-3.5 のモデルと与えるプロンプト

GPT-3.5 のモデルとして，gpt-3.5-turbo を利用した．GPT-3.5 のモデルの中で，gpt-3.5-turbo の能力が最も高いと報告されているためである [23].

また，GPT-3.5 へ与えるプロンプトを図 2 に記す．図 2 に示したプロンプトの 14，15 行目にて，メソッドの返り値を説明するように指示している．本実験で利用する FEMPDataset の等価ペア，類似ペア，不等価ペアのメソッドは必ず void 型以外の型の返り値を持つ．メソッドの返り値はそのメソッドの出力にあたるため，返り値の説明によってメソッドの機能を説明できると考えた．また，返り値の説明文は 1 文となるように限定している．これは，返り値の説明文に変数の意味やメソッドの目的といった，返り値に関係しない情報が含まれないようにするためである．

プロンプトからは 1 組の質問例と回答例を与えている．図 2 の 1 行目から 11 行目までがこれに該当する．これは，GPT-3.5 による回答の詳細さをそろえるためである．回答の詳細が揃っていない場合には，メソッドの機能的類似度を正確に算出できないと考えた．例えば，あるメソッドの説明文は，返り値の型にしか言及しておらず，別のメソッドの説明文は利用しているアルゴリズムやデータ構造まで説明していた場合，機能的類似度は正確に測定できない．したがって，回答の詳細さをそろえるため，質問例と回答例をプロンプトから与えた．

5.4 利用する Sentence-BERT のモデルとファインチューニング

Sentence-BERT のモデルには sentence-transformers/all-mpnet-base-v2[*1] というモデルをファインチューニングして利用した．ファインチューニングではバッチサイズを 48，エポック数を 20 とし，損失関数にはコサイン類似度を用いた．等価ペアのコサイン類似度が 1.0 に，類似ペアのコサイン類似度が 0.3 に，不等価ペアのコサイン類似度が 0.0 となるようにファインチューニングした．ファインチューニングに利用したデータ数は表 1 にまとめる．

表 1: Sentence-BERT のファインチューニングに利用したデータ数

ペア	学習データ数	テストデータ数
等価ペア	1,200	100
類似ペア	600	100
不等価ペア	600	100

[*1] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

5.5 RQ1 への回答

既存手法の InferCode，ASTNN と提案手法を比較する．InferCode は GitHub[*2]にある説明に従いインストールし利用した．ASTNN は GitHub[*3]にある説明に従い事前学習モデルを取得し，ファインチューニングして利用した．ASTNN のファインチューニングに利用したデータ数の内訳を表 2 に記す．

提案手法，InferCode，ASTNN の結果をそれぞれ図 3，図 4，図 5 に記す．図の横軸は，クローンか否かの判断基準となる閾値であり，縦軸は，その閾値における accuracy/ recall_E/ recall_I を表す．図 3，図 4，図 5 からわかるように，それぞれの手法の検出精度が最も良いかときの閾値は異なる．そのため本調査ではこれら 3 つの手法をそれぞれが最も良い accuracy を取るときの閾値において比較する．

表 3 にそれぞれの手法が最も良い accuracy を取るときの閾値と recall_E，recall_I を記す．表中

表 2: ASTNN のファインチューニングに利用したデータ数

ペア名	訓練データ数	検証データ数	テストデータ数
等価ペア	1,100	100	100
類似ペア	500	100	100
不等価ペア	500	100	100

表 3: 提案手法，InferCode，ASTNN の結果

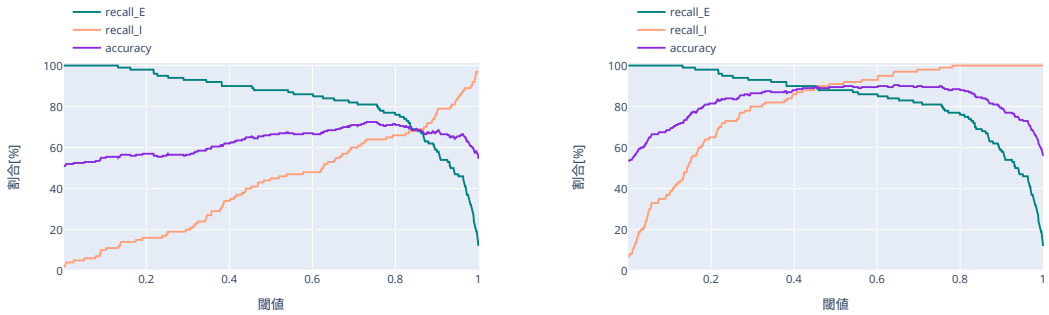
(a) 等価ペアと類似ペアに対する結果

手法	閾値	accuracy(%)	recall_E(%)	recall_I(%)
提案手法	0.802	**73.5**	74.0	**73.0**
InferCode	0.978	60.5	**85.0**	36.0
ASTNN	0.484	64.5	**85.0**	44.0

(b) 等価ペアと不等価ペアに対する結果

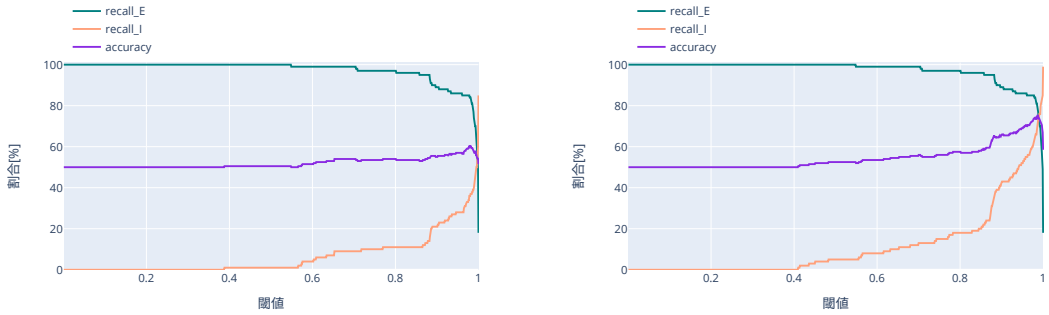
手法	閾値	accuracy(%)	recall_E(%)	recall_I(%)
提案手法	0.589	**91.0**	**87.0**	**95.0**
InferCode	0.987	75.5	78.0	73.0
ASTNN	0.484	77.5	85.0	70.0

[*2] https://github.com/bdqnghi/infercode
[*3] https://github.com/zhangj111/astnn



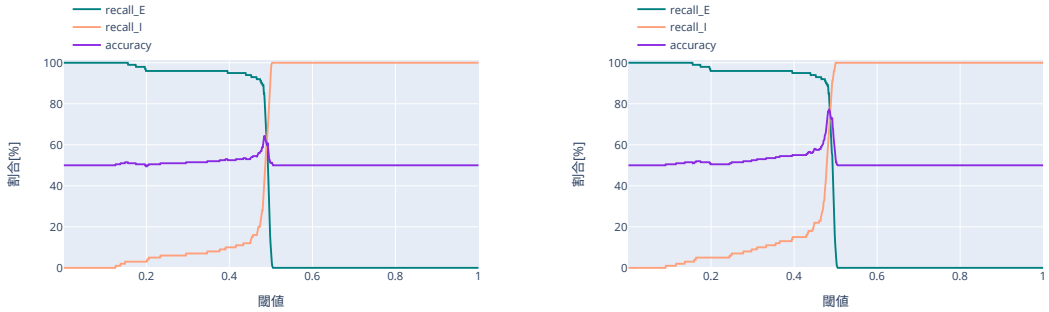
(a) 等価ペアと類似ペアを対象とした結果　(b) 等価ペアと不等価ペアを対象とした結果

図 3: 提案手法をクローン検出に活用した場合の結果



(a) 等価ペアと類似ペアを対象とした結果　(b) 等価ペアと不等価ペアを対象とした結果

図 4: InferCode をクローン検出に活用した場合の結果



(a) 等価ペアと類似ペアを対象とした結果　(b) 等価ペアと不等価ペアを対象とした結果

図 5: ASTNN をクローン検出に活用した場合の結果

の太字は，提案手法，InferCode，ASTNN の中で最も accuracy，recall_E，recall_I が良いことを意味する．まず，等価ペアと類似ペアに対する結果に着目する．accuracy は，提案手法が最も良く，InferCode に対しては 13.0%，ASTNN に対しては 9.0% 優れていることを確認した．次に，等価ペアと不等価ペアに対する結果に着目する．accuracy は提案手法が最も優れており，InferCode に対しては 15.5%，ASTNN に対しては 13.5% 優れていた．等価ペアと類似ペアを対象とした場合と，等価ペアと不等価ペアを対象とした場合の両方で，提案手法が recall_I の観点で InferCode と ASTNN を大きく

```
1 Please assess the functional similarity of the following two code snippets and provide a
   similarity score between 0 and 10.
2 A higher score indicates that the two codes are more functionally similar.
3 Output the similarity score.
4
5 % 機能の類似度を計測したいメソッドペアを与える.
```

図 6: GPT-Direct で利用するプロンプト

上回っていると確認した．このことは，提案手法は機能的に等価でないペアをクローンでないと正しく判断できたことを意味する．これまでの研究 [18] により，InferCode や ASTNN はクローンでないペアに対しても誤ってクローンと判断してしまうことが報告されており，提案手法はこの課題を大きく改善できている．提案手法が InferCode や ASTNN と比べ優れている理由への考察は，6 節にて述べる．

RQ1 への回答

等価ペアと不等価ペアに対する結果と, 等価ペアと類似ペアに対する結果から, 提案手法の accuracy が InferCode や ASTNN よりも優れており，特に recall_I の観点で InferCode や ASTNN を大きく上回ると確認した．得られた実験結果から，特に提案手法はクローン検出の観点で InferCode や ASTNN より優れていると結論づける．

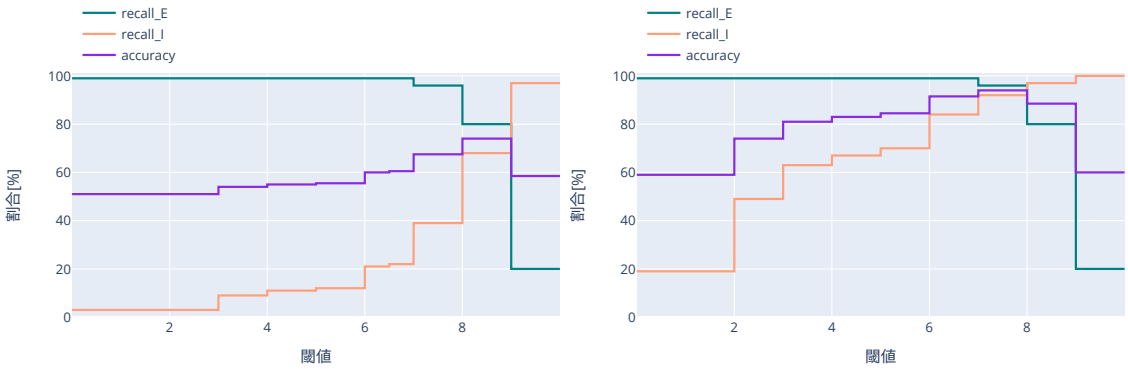
5.6 RQ2 への回答

GPT-3.5 により直接コード間の機能的類似度を算出させる手法を GPT-Direct と呼称する．GPT-3.5 へは図 6 に記した指示文と，機能的類似度を算出したいメソッドペアを与える．指示文は Dou らの研究 [32] を参考に作成した．

検出精度の比較

GPT-Direct の実験結果を図 7 に記す．RQ1 と同様に，提案手法と GPT-Direct がそれぞれ最も良い accuracy を取るときの閾値にて検出精度を比較する．なお，GPT-Direct はコード間の類似度を 0 から 10 の範囲で算出するため，閾値は 0 から 10 の範囲を 1,000 等分するように動かしている．

提案手法と GPT-Direct が最も良い accuracy を取るときの閾値と recall_E，recall_I を表 4 に記す．提案手法は GPT-Direct に対して accuracy が，等価ペアと類似ペアに対しては 0.5%，等価ペアと不等価ペアに対しては 3.0% 劣ると確認した．また，等価ペアと類似ペアを対象とした場合と，等価ペアと不等価ペアを対象とした場合の両方で，提案手法は GPT-Direct に対して recall_I の観点で優



(a) 等価ペアと類似ペアを対象とした結果　(b) 等価ペアと不等価ペアを対象とした結果

図 7: GPT-Direct をクローン検出に活用した場合の結果

れているが，recall_E の観点で劣ると確認した．これらの結果から提案手法は GPT-Direct と比べて，クローンでないペアに対して正しくクローンでないと判別できるが，クローンペアに対してもクローンでないと誤って判別してしまう傾向にあるといえる．

実行時間の比較

ソフトウェアの再利用の観点において，コードの機能に基づいた分類は重要である [33]．本評価では提案手法と GPT-Direct を用いて，コードを機能に基づき分類することを想定し，n 個のコードにおいて，すべてのペアの機能的類似度を算出するために要する時間を比較する．本評価では n の値を 2 から 10,000 まで変動させる．しかし，全ての n に対して提案手法と GPT-Direct を実行し，実行時間を計測するには非常に長い時間を要するため，現実的でない．そこで，本調査では GPT-3.5 のレスポンスに要する時間，Sentence-BERT が埋め込み表現を導出するのに要する時間，埋め込み表現のコサイ

表 4: 提案手法と GPT-Direct の実験結果

(a) 等価ペアと類似ペアに対する結果

手法	閾値	accuracy(%)	recall_E(%)	recall_I(%)
提案手法	0.802	73.5	74.0	**73.0**
GPT-Direct	8.01	**74.0**	**80.0**	68.0

(b) 等価ペアと不等価ペアに対する結果

手法	閾値	accuracy(%)	recall_E(%)	recall_I(%)
提案手法	0.589	91.0	87.0	**95.0**
GPT-Direct	7.01	**94.0**	**96.0**	92.0

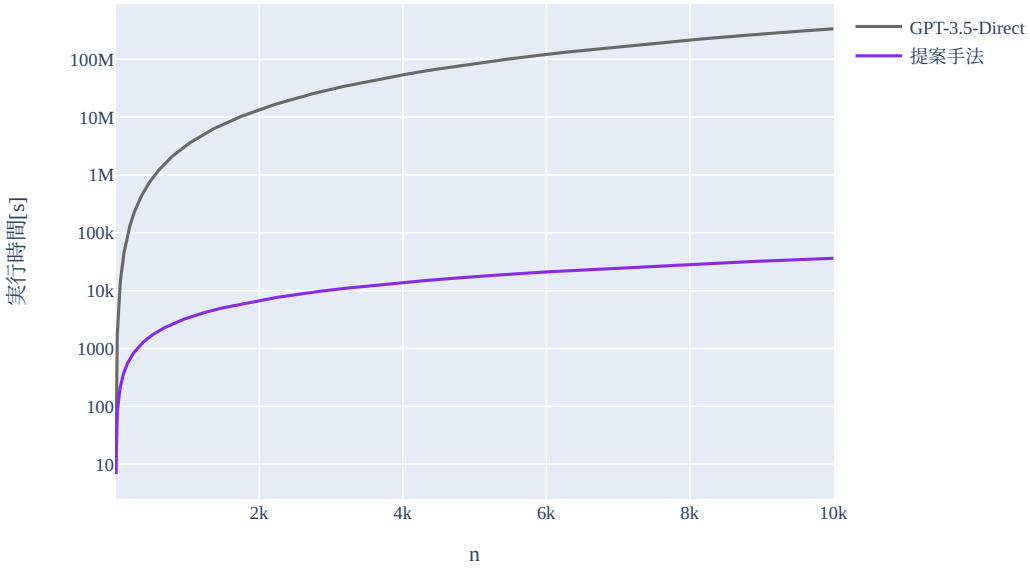


図 8: 提案手法と GPT-Direct の実行に要する時間

ン類似度を計算するのに要する時間を計測し，それらを元に提案手法と GPT-Direct が要する時間を算出する．提案手法と GPT-Direct の実行時間をそれぞれ `Time(`**提案手法**`)`，`Time(GPT-3.5-Direct)` とすると，これらはそれぞれ以下の式 (1)，式 (2) により求まる．

`Time(`**提案手法**`)`

$$\begin{aligned} &= n \times T_{gpt-propose} + n \times T_{embedding} + {}_nC_2 \times T_{cos} \\ &= n(T_{gpt-propose} + T_{embedding}) + \frac{n(n-1)}{2} T_{cos} \end{aligned} \tag{1}$$

`Time(GPT-3.5-Direct)`

$$\begin{aligned} &=_n C_2 \times T_{gpt-direct} \\ &= \frac{n(n-1)}{2} T_{gpt-direct} \end{aligned} \tag{2}$$

$T_{gpt-direct}$： GPT-Direct にて，GPT-3.5 にクエリを送信し，レスポンスを得るのに要する時間

$T_{gpt-propose}$： 提案手法にて，GPT-3.5 にクエリを送信し，レスポンスを得るのに要する時間

$T_{embedding}$： メソッドの説明文から Sentence-BERT を用いて埋め込み表現を得るのに要する時間

T_{cos}： 1 組の埋め込み表現のコサイン類似度を得るのに要する時間

実験により測定した $T_{gpt-direct}$，$T_{gpt-propose}$，$T_{embedding}$，T_{cos} を表 5 に記す．それぞれ，100 回あたりに要する時間を計測し，計測結果から 1 回あたりの所要時間を算出した．計測したこれらの値を元に，提案手法と GPT-Direct の実行に要する時間を図 8 に示す．縦軸が実行時間を，横軸が n の値を表す．縦軸は対数表示となっている．

グラフからわかるように，提案手法の方が実行時間は短い．特に，n の値が大きい場合に提案手法と GPT-Direct の実行時間の差は顕著となる．n=10,000 の条件下では，GPT-Direct ではすべてのペアの機能的類似度を算出するのに 93,601 時間 $\simeq$ 3,900 日かかる計算となり，現実的でない．しかし，提案手法であれば n=10,000 の条件下であっても，約 10 時間と現実的な時間内にすべてのペアの機能的類似度を算出できる．

RQ2 への回答

提案手法と GPT-Direct のクローン検出精度を比較したところ，accuracy の観点で提案手法の方が 0.5% - 3.0% 劣ることを確認した．また，提案手法と GPT-Direct の実行速度を比較したところ，本実験の条件下では提案手法の方が速いことを確認した．

5.7 RQ3 への回答

Chain-of-Thought を利用したプロンプトとして，以下の 2 種類のプロンプトを利用する．

表 5: 実験により計測した所要時間

$T_{gpt-direct}$	$6.74[s]$
$T_{gpt-propose}$	$3.31[s]$
$T_{embedding}$	$2.21 \times 10^{-2}[s]$
T_{cos}	$5.72 \times 10^{-5}[s]$

表 6: Chain-of-Thought を利用しない場合と，プロンプト 1，2 を利用した場合の結果

(a) 等価ペアと類似ペアに対する結果

利用したプロンプト	閾値	accuracy(%)	recall_E(%)	recall_I(%)
Chain-of-Thought を利用しない	0.802	**73.5**	74.0	**73.0**
プロンプト 1	0.457	63.5	**85.0**	42.0
プロンプト 2	0.747	68.0	65.0	71.0

(b) 等価ペアと不等価ペアに対する結果

利用したプロンプト	閾値	accuracy(%)	recall_E(%)	recall_I(%)
Chain-of-Thought を利用しない	0.589	**91.0**	87.0	**95.0**
プロンプト 1	0.457	85.0	85.0	85.0
プロンプト 2	0.457	90.5	**90.0**	91.0

プロンプト 1： メソッドの境界値を説明させ，その後にメソッドの返り値を説明させる．

プロンプト 2： メソッドへ与えた引数の意味を説明させた後に，メソッドを利用する目的について説明させ，その後にメソッドの返り値を説明させる．

プロンプト 1，2 として実験で利用したプロンプトは本論文の末尾に設けた付録に記す．メソッド間の機能的類似度の算出には，それぞれのプロンプトで得られたメソッドの返り値に関する説明部のみを利用する．

プロンプト 1，プロンプト 2 を利用した場合と，Chain-of-Thought を利用しない場合で accuracy が最も良い値を取るときの閾値，recall_E，recall_I を表 6 にまとめる．表中の太字は，accuracy，recall_E，recall_I が最も良いことを意味する．等価ペアと類似ペアを対象とした場合と，等価ペアと不等価ペアを対象とした場合の両方で，プロンプト 1，2 による Chain-of-Thought を利用した場合には，利用しなかった場合と比べ accuracy はかえって低下すると確認した．Chain-of-Thought により accuracy が低下する理由は 6 節にて考察する．

RQ3 への回答

Chain-of-Thought を取り入れる前後でのクローン検出精度を比較した結果，本実験で利用したプロンプトでは検出精度の向上は確認されなかった．

6 考察

実験で得られた結果に対する考察や，提案手法における処理が有効だったかの調査結果を記す．提案手法，InferCode，ASTNN，GPT-Direct を比較する場合には，各手法の閾値としてそれぞれが最も良い accuracy を取るときの閾値を用いる．

6.1 提案手法の Step-2 における変数名の置換は有効か？

提案手法では，GPT-3.5 が生成した説明文に含まれる変数名を`$VAR`で置換し正規化していた．この目的は，説明文中に含まれる変数名が，コード間の機能的類似度を算出する上でノイズになると考えたためである．

変数名を置換する場合と置換しない場合を比較し，提案手法における文字列の置換が有効だったか調査する．変数名を書き換えた場合と書き換えなかった場合の accuracy の最大値と，その時の閾値，recall_E，recall_I を表 7 にまとめる．表からわかるように，変数名を置換した方が accuracy が良い結果が得られた．このことは，提案手法において変数名がコード間の機能的類似度を計測する上でノイズとなっていたことを意味する．したがって，Step-2 における変数名の置換は，提案手法がより正確に機能的類似度を算出する上で有効であると結論づける．

表 7: 変数名を書き換える場合と書き換えなかった場合の検出精度の違い

(a) 等価ペアと類似ペアに対する結果

検出方法	閾値	accuracy(%)	recall_E(%)	recall_I(%)
変数名を書き換える場合	0.802	73.5	74.0	73.0
変数名を書き換えない場合	0.724	70.5	77.0	64.0

(b) 等価ペアと不等価ペアに対する結果

検出方法	閾値	accuracy(%)	recall_E(%)	recall_I(%)
変数名を書き換える場合	0.589	91.0	87.0	95.0
変数名を書き換えない場合	0.516	90.0	91.0	89.0

6.2 提案手法は InferCode や ASTNN，GPT-Direct と比べ，構文的類似性が機能的類似性に与える影響を抑えられているか？

構文的類似性が提案手法，ASTNN，InferCode，GPT-Direct が算出する機能的類似性に与える影響を調査する．本節では，構文的類似性をペア間のステートメント数と分岐数の観点から評価する．メソッド間のステートメント数や分岐数の差と，各手法が算出した機能的類似度の相関を算出し，負の相関，あるいは正の相関が強い場合に機能的類似度が構文的類似度から受けている影響が強いと判断する．

まず，ペア間のステートメント数の差と機能的類似度の相関に着目する．表 8 に，ペア間のステートメント数の差と機能的類似度に関するスピアマンの順位相関係数および有意性検定の結果である p 値を示す．有意水準は 0.01 とする．表中の太文字は，p 値が有意水準を下回っていることを意味する．等価ペア，類似ペア，不等価ペアのすべてを対象とした場合において，提案手法による機能的類似度とペア間のステートメント数の差に関する相関が最も弱いと確認した．また，ASTNN や InferCode では，等価ペア，類似ペア，不等価ペアのすべてで p 値が有意水準を下回ることを，GPT-Direct では，等価ペアと類似ペアで有意水準を下回ることを確認した．それに対し，提案手法は等価ペア，類似ペア，不等価ペアのすべてで，p 値が有意水準を下回らなかった．このことから，ASTNN や InferCode，GPT-Direct は，提案手法と比べてペア間のステートメント数に差がある場合に低い機能的類似度を算出する傾向にあるといえる．

表 8: 機能的類似度とペア間のステートメント数の差に関するスピアマンの順位相関係数

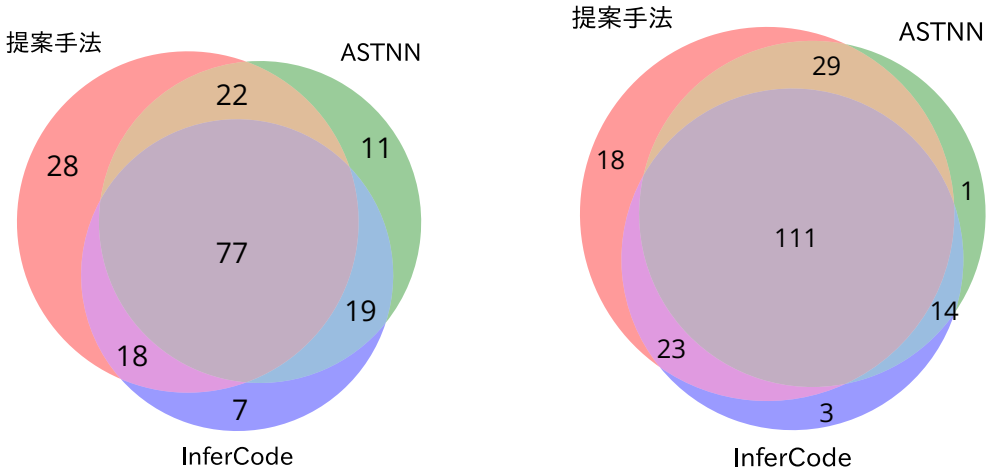
対象	手法	相関係数	p 値
等価ペア	提案手法	-0.26	0.010
等価ペア	ASTNN	-0.34	**0.00052**
等価ペア	InferCode	-0.46	$\mathbf{1.6 \times 10^{-6}}$
等価ペア	GPT-Direct	-0.46	$\mathbf{1.3 \times 10^{-6}}$
類似ペア	提案手法	-0.11	0.26
類似ペア	ASTNN	-0.61	$\mathbf{1.1 \times 10^{-11}}$
類似ペア	InferCode	-0.31	**0.0016**
類似ペア	GPT-Direct	-0.34	**0.00041**
不等価ペア	提案手法	-0.18	0.079
不等価ペア	ASTNN	-0.60	$\mathbf{6.1 \times 10^{-11}}$
不等価ペア	InferCode	-0.56	$\mathbf{1.3 \times 10^{-9}}$
不等価ペア	GPT-Direct	-0.23	0.018

次に，ペア間の分岐数の差と機能的類似度の相関に着目する．表 9 に，ペア間の分岐数の差と機能的類似度に関するスピアマンの順位相関係数および有意性検定の結果である p 値を示す．表の見方は表 8 と同様である．等価ペア，類似ペア，不等価ペアのすべてを対象とした場合において，提案手法による機能的類似度とペア間の分岐数の差に関する相関が最も弱いと確認した．また，ASTNN や InferCode では，等価ペア，類似ペア，不等価ペアのすべてで p 値が有意水準を下回ることを，GPT-Direct では等価ペアと類似ペアで有意水準を下回ることを確認した．それに対し，提案手法は等価ペア，類似ペア，不等価ペアのすべてで，p 値が有意水準を下回らなかった．このことから，ASTNN や InferCode, GPT-Direct は，提案手法と比べてペア間の分岐数に差がある場合に低い機能的類似度を算出する傾向にあるといえる．

以上の結果から，ペア間の構文的類似性をステートメント数と分岐数の差から計測した場合，提案手法は ASTNN や InferCode，GPT-Direct と比べ，構文的類似性が機能的類似性に与える影響を抑えられていたといえる．この理由は，ASTNN や InferCode，GPT-Direct が機能的類似性を算出に，2 つのコードを直接受け取るのに対し，提案手法では自然言語によるコードの説明文を受け取っているためだと考える．すなわち，提案手法では機能的類似性の算出にコードを直接的に利用しないため，構文的類似性が機能的類似性に与える影響を他の手法と比べて抑えられたと考える．

表 9: 機能的類似度とペア間の分岐数の差に関するスピアマンの順位相関係数

対象	手法	相関係数	p 値
等価ペア	提案手法	-0.27	0.010
等価ペア	ASTNN	-0.35	**0.00035**
等価ペア	InferCode	-0.44	$\mathbf{6.0 \times 10^{-6}}$
等価ペア	GPT-Direct	-0.45	$\mathbf{2.2 \times 10^{-6}}$
類似ペア	提案手法	-0.026	0.80
類似ペア	ASTNN	-0.49	$\mathbf{1.8 \times 10^{-7}}$
類似ペア	InferCode	-0.31	**0.0019**
類似ペア	GPT-Direct	-0.32	**0.0013**
不等価ペア	提案手法	-0.084	0.40
不等価ペア	ASTNN	-0.51	$\mathbf{6.6 \times 10^{-8}}$
不等価ペア	InferCode	-0.35	**0.00041**
不等価ペア	GPT-Direct	-0.14	0.17



(a) 等価ペアと類似ペアに対するベン図　　(b) 等価ペアと不等価ペアに対するベン図

図 9: 提案手法と InferCode，ASTNN がそれぞれクローン/クローンでないと正しく検出できたペアのベン図

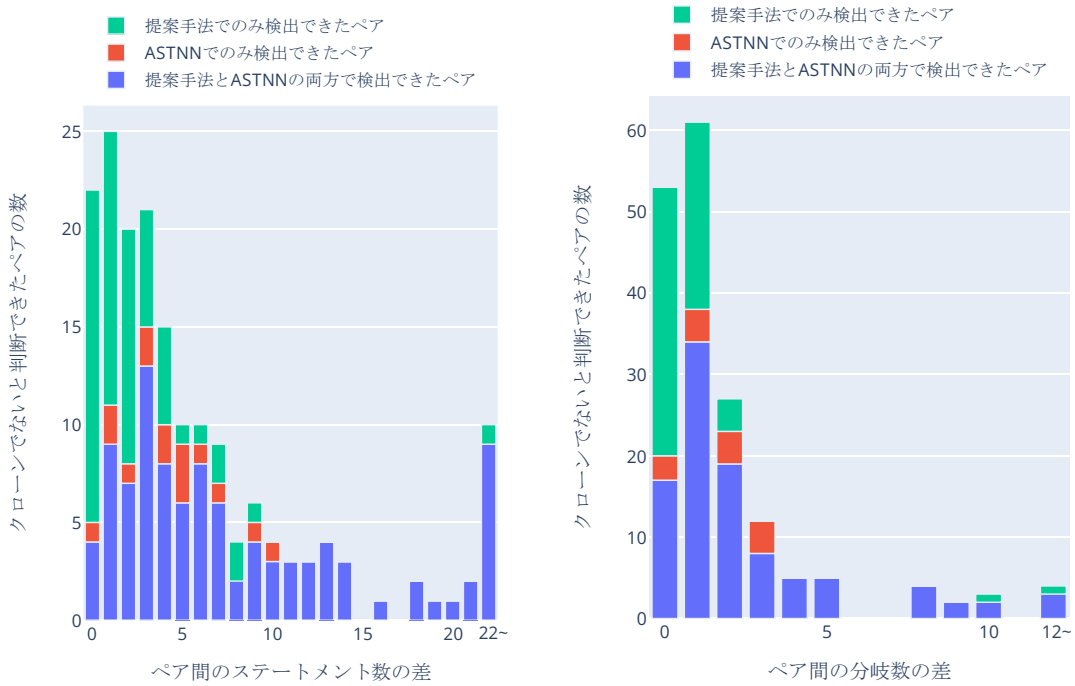
6.3 なぜ提案手法は ASTNN や InferCode よりもクローン検出精度が良かったのか？

提案手法が InferCode や ASTNN よりクローン検出精度が優れていた理由を考察する．提案手法と InferCode，ASTNN がそれぞれクローン/クローンでないと正しく検出できたペアのベン図を図 12 に記す．図中の円は，それぞれの手法が正しく検出できたペアの集合を表している．

5.5 節の結果より，提案手法は ASTNN や InferCode と比べて recall_I の観点で大きく優れていると確認した．これは，提案手法が ASTNN や InferCode と比べ，クローンでないペアを正しくクローンでないと判断できた割合が高いことを意味する．この理由は，6.2 項で示した結果にあると考える．6.2 項より，ASTNN や InferCode は，ペア間のステートメント数や分岐数の差と機能的類似度の負の相関があるとわかった．このことから ASTNN や InferCode はペア間のステートメント数や分岐数に差がない場合，クローンでないペアに対しても高い機能的類似度を算出したと仮説を立てた．

実験により，この仮説が正しいか検証する．クローンでないペア，すなわち，類似ペアと不等価ペアに着目し，提案手法ではクローンでないと正しく判断できるが ASTNN や InferCode では誤ってクローンと判断してしまうペアが，ペア間のステートメント数や分岐数の差が小さい範囲に多く分布するかの調査により仮説を検証できると考えた．

まず，提案手法と ASTNN の実験結果に着目する．図 10 に，クローンでないと正しく判断できたペアを縦軸に，ペア間のステートメント数の差を横軸にとった積み上げ棒グラフを示す．棒グラフの緑色の部分が提案手法のみクローンでないと正しく判断できたペアを，赤色の部分が ASTNN のみがクローンでないと正しく判断できたペアを，青色の部分が提案手法と ASTNN の両方でクローンでないと正しく判断できたペアを意味する．なお，グラフの見やすさのため，ステートメント数の差が 22 以上のペ



(a) 横軸にステートメント数の差をとった場合　(b) 横軸に分岐数の差をとった場合

図 10: 提案手法と ASTNN がクローンでないと正しく判断できた類似ペア，不等価ペアの個数の分布

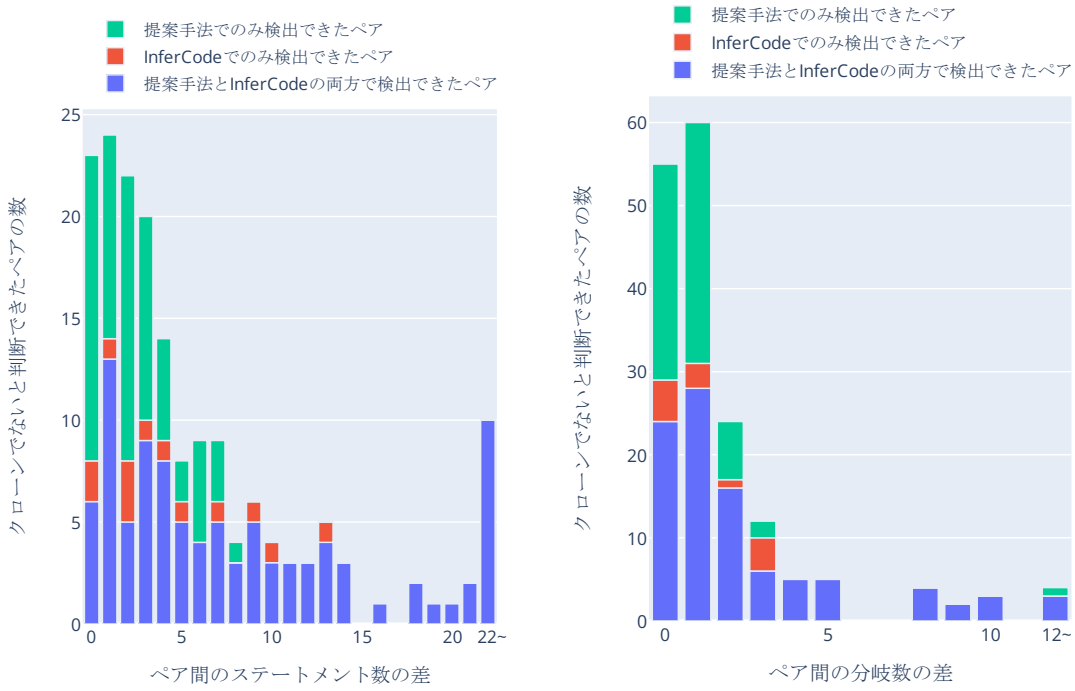
アと，分岐数の差が 12 以上のペアはまとめて表示する．結果を見ると，提案手法でのみクローンでないと正しく判断できたペアはステートメント数や分岐数の差が小さい範囲に分布しているとわかる．

次に，提案手法と InferCode の実験結果に着目する．提案手法と ASTNN を比較したときと同様に，クローンでないと判断できたペア数を縦軸に，ステートメント数や分岐数の差を横軸にとった積み上げ棒グラフを図 11 に示す．提案手法と ASTNN を比較した場合と同様に，提案手法でのみクローンでないと正しく判断できたペアはステートメント数や分岐数の差が小さい範囲に分布しているとわかる．

これらの実験結果から，本節で立てた ASTNN や InferCode はペア間のステートメント数や分岐数に差がない場合，クローンでないペアに対しても高い機能的類似度を付与する傾向にあるという仮説は正しいといえる．得られた結果から，提案手法が ASTNN や InferCode と比べクローンの検出精度が良い理由は，ペア間のステートメント数や分岐数の差が小さい，クローンでないペアに対して機能的類似性を正しく計測し，低い類似度を算出できているためだと結論づける．

6.4　提案手法と GPT-Direct で正しく検出できるペアにはどのような違いがあるか？

提案手法と GPT-Direct がクローン/クローンでないと正しく検出できたペアのベン図を図 12 に記す．等価ペアと類似ペアを対象とした場合と，等価ペアと不等価ペアを対象とした場合で，提案手法でのみ検出できたペアは合計 36 個，GPT-Direct でのみ検出できたペアは合計 46 個であった．これらの

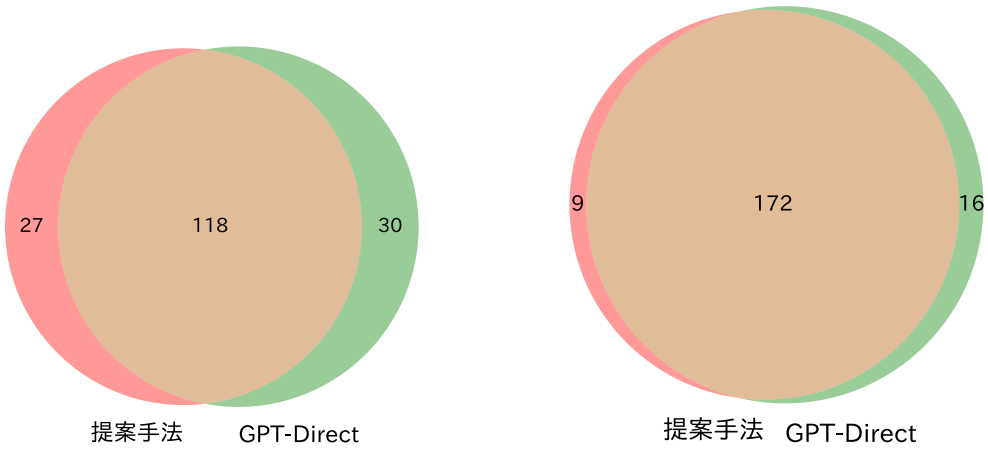


(a) 横軸にステートメント数の差をとった場合　(b) 横軸に分岐数の差をとった場合

図 11: 提案手法と InferCode がクローンでないと正しく判断できた類似ペア，不等価ペアの個数の分布

ペアを目視で確認し，提案手法と GPT-Direct で正しく検出できるペアにどのような特徴があるか調査した．その結果，提案手法において，メソッドの機能を適切に表現できていないメソッド名がコードの説明文に影響を与え，提案手法が機能等価にも関わらず低い機能的類似度を付与している例を発見した．

図 13 にそのペアを記す．説明文 A，B はそれぞれ提案手法において GPT-3.5 により生成されたメソッド A，B の説明文である．メソッド A，メソッド B は，どちらも第一引数に与えられた `byte` 型の配列の先頭部分が第二引数に与えられた `byte` 型の配列と一致するか検査し，一致する場合には `true`，一致しない場合には `false` を返す．説明文 A では，誤ってこのメソッドを説明している．検査するのは第一引数の先頭部分と第二引数であるはずだが，説明文 A ではメソッド A を，第一引数の `header` と第二引数の `suggestedEncodingBytes` が一致するか検査するメソッドと説明している．これは，メソッド A のメソッド名が `matches` であるためと考える．メソッド A の機能は配列の前方が一致しているかの判定であるにも関わらず，メソッド名が `matches` と配列全体が一致するか判定するようなメソッド名であるため，誤った説明文が生成されてしまった．一方で，説明文 B ではメソッド B を，第一引数の `checkMe` が第二引数の `maybePrefix` から始まるか，すなわち，第一引数の `checkMe` の先頭部分が第二引数の `maybePrefix` と一致するか検査するメソッドと，正しく説明出来ている．メソッド B のメソッド名が正しくメソッドの機能を表現できていたため，正しい説明文を生成できたと考える．



(a) 等価ペアと類似ペアに対するベン図　　(b) 等価ペアと不等価ペアに対するベン図

図 12: 提案手法と GPT-Direct がそれぞれクローン/クローンでないと正しく検出できたペアのベン図

```
boolean matches(byte[] header, byte[]
suggestedEncodingBytes){
  if (header.length < suggestedEncodingBytes.length)
    return false;
  for (int i=0; i < suggestedEncodingBytes.length; i++)
    if (header[i] != suggestedEncodingBytes[i])
      return false;
  return true;
}
```

メソッドA

```
boolean startsWith(byte[] checkMe, byte[] maybePrefix){
  int cm_len=checkMe.length;
  int mp_len=maybePrefix.length;
  if (cm_len < mp_len)
    return false;
  for (int i=0; i < mp_len; ++i)
    if (checkMe[i] != maybePrefix[i])
      return false;
  return true;
}
```

メソッドB

The return value of the program is 'true' if the 'header' array matches the 'suggestedEncodingBytes' array, and 'false' otherwise.

説明文A

The return value of the program is 'true' if the 'checkMe' array starts with the 'maybePrefix' array, and 'false' otherwise.

説明文B

図 13: 機能等価だが，提案手法が低い類似度を算出したペアの例

そのため，メソッド A に対して不適切な説明文を生成したため，これらのペアは機能等価であるにも関わらず，クローンでないと判断されてしまった．したがって，提案手法によって機能的類似性を正確に計測するためには，メソッド名を適切に付与する必要があると考える．

6.5 RQ3 にて Chain-of-Thought を取り入れた場合に精度が悪化した理由は何か？

プロンプト 1 による Chain-of-Thought を取り入れることにより機能的類似度が悪化する例を図 14 に示す．図 14 に記したメソッドペアは機能的に等価なメソッドペアである．Chain-of-Thought を利用しない場合の提案手法による機能的類似度は 1.0 であったが，プロンプト 1 による Chain-of-Thought を取り入れることで機能的類似度は 0.53 にまで悪化した．

```
boolean _part3Compare(byte[] a,byte[] b){
  for (int i=0; i < 16; i++) {
    if (a[i] != b[i + 4]) {
      return false;
    }
  }
  return true;
}
```

メソッドA

```
boolean _part3Compare(byte[] a,byte[] b){
  for (int i=0; i < 16; i++)
    if (a[i] != b[i + 4])
      return false;
  return true;
}
```

メソッドB

Q1. 境界値について教えてください.
The boundary values for the given method involve the byte arrays 'a' and 'b', where the length of both arrays should be at least 20 bytes to avoid an IndexOutOfBoundsException, and the values of 'a' and 'b' should be equal for the first 16 elements starting from index 0 and index 4 respectively.

Q2. メソッドの返り値を説明してください.
The return value of the program is true if the first 16 elements of byte array 'a' are equal to the elements of byte array 'b' starting from index 4, otherwise it returns false.

説明文A

Q1. 境界値について教えてください.
The boundary values for the given method involve the byte arrays 'a' and 'b', where the values of 'a' and 'b' are compared element-wise starting from index 4 of 'b' and if any element of 'a' is not equal to the corresponding element of 'b', the method returns false, otherwise it returns true.

Q2. メソッドの返り値を説明してください.
The return value of the program is true if all corresponding elements of the byte arrays 'a' and 'b' starting from index 4 of 'b' are equal, otherwise it returns false.

説明文B

図 14: Chain-of-Thought により機能的類似度が悪化する例

図 14 に記した説明文 A，B は，それぞれ GPT-3.5 によって生成されたメソッド A，B の説明文である．メソッドの境界値について説明をさせ，次にメソッドの返り値を説明させている．メソッド A，B はどちらも for 文の条件式の部分に 16 を境界値に持つ．説明文 A を見ると，境界値について説明をさせた際に，この 16 という境界値に関する説明文が生成された．したがって，その後にメソッドの返り値を説明させた際にも，16 という境界値に着目した説明が生成された．一方で，説明文 B を見ると，境界値について説明させた際に，16 という境界値に触れた説明が生成されなかった．そのため，その後にメソッドの返り値を説明させた際にも，16 という境界値に着目した説明が生成されなかった．よって，16 という境界値に言及しているかという点で説明文 A と説明文 B の意味的な類似度が低下し，メソッド A，B の機能的類似度は悪化した．

このように，プロンプト 1，2 による Chain-of-Thought を利用することで，前の質問に対する回答の差異が，最終的なメソッドの説明文に入り込み，機能的類似性を計測する上で悪影響を及ぼしたと考察する．

7　妥当性の脅威

提案手法をクローン検出ツールとして評価した際に，FEMPDataset を実験対象として利用した．FEMPDataset には，`java.lang` 及び `java.util` パッケージ以外の参照型を含んだメソッドは含まれない．`java.lang` や `java.util` 以外の参照型を含むメソッドペアを実験対象とした場合，異なる実験結果が得られる可能性がある．また，FEMPDataset に含まれる言語はすべて Java 言語で記述されている．異なる言語で記述されたデータセットを対象に実験をした場合，本研究とは異なる結果が得られる可能性がある．

GPT-3.5 は，同一のプロンプトであっても，実行毎に異なる回答を出力する可能性がある．したがって，本研究と同じプロンプトと同じモデルを利用した場合でも，本研究とは異なる実験結果が得られる可能性がある．

本研究では，提案手法との比較対象として ASTNN を利用した．ASTNN のファインチューニングに利用した学習用データ数は，提案手法での Sentence-BERT のファインチューニングに利用したデータ数と同じになるようにした．学習用データ数をより多くした場合には，本研究で得られた結果とは異なる結果が得られる可能性がある．

考察にて，コード間の構文的類似性をステートメント数と分岐数の観点から評価した．しかし，構文的類似性はこれら以外にも，データ依存やコントロール依存の観点や，コードのトークンの並びといった観点からの評価が考えられる．異なる観点から評価した場合，異なる考察結果が得られる可能性がある．

8 おわりに

本研究では, コード間の機能的類似性の定量化を目的とした手法を提案した．提案手法では, GPT-3.5 にコードの機能を説明させ，出力させた説明文の意味的な類似度を Sentence-BERT により計測する．コードから直接機能的類似度を算出するのではなく，コードの説明文を介して機能的類似度を計測するため，構文的類似性が機能的類似性に与える影響を抑えられると考えた．

提案手法をクローン検出ツールとして評価したところ，既存の InferCode や ASTNN よりも高い精度でクローンを検出できると確認した．また，GPT-3.5 に直接コード間の類似度を算出させた場合と比較したところ，検出精度では提案手法の方が 0.5%-3.0% 劣るが，実行速度では提案手法の方が速い結果が得られた．また，構文的類似性が提案手法や GPT-Direct，ASTNN，InferCode の算出する機能的類似度に与える影響を調査し，ペア間の構文的類似性をステートメント数と分岐数から計測した場合には提案手法はその他の手法と比べ，構文的類似性が機能的類似性に与える影響を抑えられていると確認した．

今後の課題について述べる．本評価では，GPT-3.5 に与えるプロンプトを提案手法と GPT-Direct でそれぞれ 1 個づつしか試していない．異なるプロンプトを与えた場合には異なる実験結果が得られる可能性があり，より多くのプロンプトを試す必要がある．また，実験対象として FEMPDataset を利用した．FEMPDataset に含まれるメソッドは，返り値が void 以外であるという特徴を持つ．本研究では GPT-3.5 にメソッドの返り値を説明させているが，void 型を返り値の型に持つメソッドに対してはメソッドの返り値以外の観点から機能を説明する必要がある．今後は実験対象を拡張し，void 型を返り値の型に持つメソッドに対しても提案手法は適用可能であるか調査する必要がある．

謝辞

本研究をするにあたり，多くの方々に多大なるご支援を賜りました．

楠本真二教授には，本研究を行うにあたり，随所での的確なご指導を賜りました．特に，中間報告会や修士論文発表会の練習では，自分では気づけなかった観点からのご指摘を数多くいただき，本研究をより良い研究へ昇華できたと感じております．

肥後芳樹教授には，学部の 4 年次から現在に至るまで，担当教員としてお世話になりました．論文の添削や学会発表の準備にて熱心なご指導をいただきました．研究で行き詰まった際には的確な助言と熱心な議論により解決へと導いてくださいました．

柗本真佑助教には，本研究に関する適切な助言を数多く賜りました．また，研究室内にて数々の愉快なイベントを企画していただき，気分をリフレッシュして楽しく研究活動を続けられました．

事務補佐員の橋本美砂子氏には，研究活動に関わる様々な手続きをしていただきお世話になりました．普段の生活においても気さくに声をかけていただき，励ましていただきました．

また，楠本研究室の先輩，同期，後輩の皆様には，研究活動に限らず，日常の生活のあらゆる面でお世話になりました．

最後に，両親と祖父母には日常生活を献身的に支えていただきました．

本研究をするにあたりお世話になったすべての皆様に深謝いたします．

参考文献

[1] Zhao,G. and Huang,J.: DeepSim: Deep Learning Code Functional Similarity, in *Proc. Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 141–151 (2018).

[2] Roy,C.K., Cordy,J.R. and Koschke,R.: Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach, *Trans. Science of Computer Programming*, Vol.74, No.7, pp. 470–495 (2009).

[3] Kamiya,T., Kusumoto,S. and Inoue,K.: CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code, *Trans. Software Engineering*, Vol.28, No.7, pp. 654–670 (2002).

[4] Holmes,R. and Murphy,G.: Using Structural Context to Recommend Source Code Examples, in *Proc. International Conference on Software Engineering*, pp. 117–125 (2005).

[5] Keivanloo,I., Rilling,J. and Zou,Y.: Spotting Working Code Examples, in *Proc. International Conference on Software Engineering*, pp. 664–675 (2014).

[6] DiGrazia,L. and Pradel,M.: Code Search: A Survey of Techniques for Finding Code, *Trans. Computing Surveys*, Vol.55, No.11, pp. 1–31 (2023).

[7] Juergens,E., Deissenboeck,F., Hummel,B. and Wagner,S.: Do code clones matter?, in *Proc. International Conference on Software Engineering*, pp. 485–495 (2009).

[8] Mehrotra,N., Agarwal,N., Gupta,P., Anand,S., Lo,D. and Purandare,R.: Modeling Functional Similarity in Source Code With Graph-Based Siamese Networks, *Trans. Software Engineering*, Vol.48, No.10, pp. 3771–3789 (2022).

[9] Baker,B.S.: A Program for Identifying Duplicated Code, in *Proc. Computing Science and Statistics*, pp. 49–49 (1992).

[10] Baker,B.S.: Parameterized Pattern Matching: Algorithms and Applications, *Trans. Journal of Computer and System Sciences*, Vol.52, No.1, pp. 28–42 (1996).

[11] Jiang,L., Misherghi,G., Su,Z. and Glondu,S.: DECKARD: Scalable and Accurate Tree-Based Detection of Code Clones, in *Proc. International Conference on Software Engineering*, pp. 96–105 (2007).

[12] Li,Z., Lu,S., Myagmar,S. and Zhou,Y.: CP-Miner: A Tool for Finding Copy-paste and Related Bugs in Operating System Code, in *Proc. Symposium on Operating Systems Design & Implementation*, pp. 289–302 (2004).

[13] Sajnani,H., Saini,V., Svajlenko,J., Roy,C.K. and Lopes,C.V.: SourcererCC: Scaling Code Clone Detection to Big-Code, in *Proc. International Conference on Software Engineering*, pp. 1157–1168 (2016).

[14] White,M., Tufano,M., Vendome,C. and Poshyvanyk,D.: Deep learning code fragments for code clone detection, in *Proc. International Conference on Automated Software Engineering*, pp. 87–98 (2016).

[15] Yang,J., Hotta,K., Higo,Y., Igaki,H. and Kusumoto,S.: Classification Model for Code Clones Based on Machine Learning, Vol.20, No.4, pp. 1095–1125 (2015).

[16] Bui,N.D., Yu,Y. and Jiang,L.: InferCode: Self-Supervised Learning of Code Representations by Predicting Subtrees, in *Proc. International Conference on Software Engineering*, pp. 1186–1197 (2021).

[17] Zhang,J., Wang,X., Zhang,H., Sun,H., Wang,K. and Liu,X.: A Novel Neural Source Code Representation Based on Abstract Syntax Tree, in *Proc. International Conference on Software Engineering*, pp. 783–794 (2019).

[18] 芳樹肥後：自動テスト生成技術を利用した機能等価メソッドデータセットの構築, ソフトウェアエンジニアリングシンポジウム, pp. 30–38 (2023).

[19] Reimers,N. and Gurevych,I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, in *Proc. Conference on Empirical Methods in Natural Language Processing*, pp. 3982–3992 (2019).

[20] Iyer,S., Konstas,I., Cheung,A. and Zettlemoyer,L.: Summarizing Source Code using a Neural Attention Model, in *Proc. Annual Meeting of the Association for Computational Linguistics*, pp. 2073–2083 (2016).

[21] 鶴智秋：深層学習を用いた意味的コードクローン検出手法の評価：SemanticCloneBench を題材として (2023).

[22] Liu,Y., Deng,G., Xu,Z., Li,Y., Zheng,Y., Zhang,Y., Zhao,L., Zhang,T. and Liu,Y.: Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study, *ArXiv*, Vol. abs/2305.13860, (2023).

[23] Ye,J., Chen,X., Xu,N., Zu,C., Shao,Z., Liu,S., Cui,Y., Zhou,Z., Gong,C., Shen,Y., Zhou,J., Chen,S., Gui,T., Zhang,Q. and Huang,X.: A Comprehensive Capability Analysis of GPT-3 and GPT-3.5 Series Models, *ArXiv*, Vol. abs/2303.10420, (2023).

[24] Devlin,J., Chang,M.-W., Lee,K. and Toutanova,K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2019).

[25] Rogers,A., Kovaleva,O. and Rumshisky,A.: A Primer in BERTology: What We Know About How BERT Works, *Trans. Association for Computational Linguistics*, Vol.8, pp. 842–866 (2021).

[26] Bromley,J., Guyon,I., LeCun,Y., Säckinger,E. and Shah,R.: Signature Verification Using a "Siamese" Time Delay Neural Network, in *Proc. International Conference on Neural Information Processing Systems*, pp. 737–744 (1993).

[27] Wei,J., Wang,X., Schuurmans,D., Bosma,M., Chi,hsin E.H., Xia,F., Le,Q. and Zhou,D.: Chain of Thought Prompting Elicits Reasoning in Large Language Models, *ArXiv*, Vol. abs/2201.11903, (2022).

[28] Fu,Y., Peng,H.-C., Sabharwal,A., Clark,P. and Khot,T.: Complexity-Based Prompting for Multi-Step Reasoning, *ArXiv*, Vol. abs/2210.00720, (2022).

[29] Li,Y., Lin,Z., Zhang,S., Fu,Q., Chen,B., Lou,J.-G. and Chen,W.: On the Advance of Making Language Models Better Reasoners, *ArXiv*, Vol. abs/2206.02336, (2022).

[30] Zheng,L., Chiang,W.-L., Sheng,Y., Zhuang,S., Wu,Z., Zhuang,Y., Lin,Z., Li,Z., Li,D., Xing,E.P., Zhang,H., Gonzalez,J. and Stoica,I.: Judging LLM-as-a-judge with MT-Bench and Chatbot Arena, *ArXiv*, Vol. abs/2306.05685, (2023).

[31] IJaDataset 2.0, `https://onedrive.live.com/?authkey=%21ABQCBsNwiW9SRjo&id=8BFCB70AA333DB15%21260599&cid=8BFCB70AA333DB15&parId=root&parQt=sharedby&o=OneUp`.

[32] Dou,S., Shan,J., Jia,H., Deng,W., Xi,Z., He,W., Wu,Y., Gui,T., Liu,Y. and Huang,X.: Towards Understanding the Capability of Large Language Models on Code Clone Detection: A Survey, *ArXiv*, Vol. abs/2308.01191, (2023).

[33] Mou,L., Li,G., Zhang,L., Wang,T. and Jin,Z.: Convolutional neural networks over tree structures for programming language processing, in *Proc. AAAI Conference on Artificial Intelligence*, pp. 1287–1293 (2016).

付録

Chain-of-Thought に利用したプロンプト

Listing1: プロンプト 1

```
# 質問例1
Please tell me the boundary values of the following code in one sentence.
'''
double clamp(double x,double min,double max){
  if (x < min) {
    return min;
  }
  if (x > max) {
    return max;
  }
  return (float)x;
}
'''
# 回答例1
The boundary values for the given method involve the minimum and maximum values for the parameters that can be passed, including the double 'x', 'min', and 'max'.

# 質問例2
Explain the return value of the following program in one sentence.
If your answer includes the name of a variable inside the program, enclose the name of the variable in single quotation marks. Do not include any information other than the return value, such as the process flow, in your answer.
'''
double clamp(double x,double min,double max){
  if (x < min) {
    return min;
  }
  if (x > max) {
    return max;
  }
  return (float)x;
}
'''
# 回答例2
The return value of the program is the value of the double 'x' constrained within the range defined by 'min' as the lower bound and 'max' as the upper bound, ensuring it does not go below 'min' or exceed 'max'.

# 質問1
Explain the return value of the following program in one sentence.
Do not include any information other than the return value, such as the process flow, in your answer.
% 境界値を説明させたいメソッドを与える

# 質問2
Explain the return value of the following program in one sentence. If your answer includes the name of a variable inside the program, enclose the name of the variable in single quotation marks. Do not include any information other than the return value, such as the process flow, in your answer.
% 返り値を説明させたいメソッドを与える
```

Listing2: プロンプト 2

```
# 質問例1
Please describe the arguments given to the following methods in one sentence.
'''
double clamp(double x,double min,double max){
  if (x < min) {
    return min;
  }
  if (x > max) {
    return max;
  }
  return (float)x;
}
'''

# 回答例1
The clamp method takes three double-precision floating-point arguments (x, min, and max) and returns min if x is less than min, max if x is greater than max, and x itself otherwise, effectively constraining the value of x within the specified range.

# 質問例2
What is the purpose of the following method?
'''
double clamp(double x,double min,double max){
```

```
    if (x < min) {
      return min;
    }
    if (x > max) {
      return max;
    }
    return (float)x;
  }
  ```
  # 回答例2
  The purpose of the clamp method is to restrict the input value x to be within the specified range defined by the min and max parameters. If the value of x is less than min, it returns min. If x is greater than max, it returns max. Otherwise, if x is already within the specified range, it returns x. This function effectively ensures that the returned value is always within the given minimum and maximum bounds.

  # 質問例3
  Explain the return value of the following program in one sentence. If your answer includes the name of a variable inside the program, enclose the name of the variable in single quotation marks. Do not include any information other than the return value, such as the process flow, in your answer.
  ```
  double clamp(double x,double min,double max){
    if (x < min) {
      return min;
    }
    if (x > max) {
      return max;
    }
    return (float)x;
  }
  ```

  # 回答例3
  The return value of the given program is the clamped version of the input parameter 'x' within the range defined by 'min' and 'max'.

  # 質問1
  Please describe the arguments given to the following methods in one sentence.
  % 説明させたいメソッドを与える

  # 質問2
  What is the purpose of the following method?
  % 説明させたいメソッドを与える

  # 質問3
  Explain the return value of the following program in one sentence. If your answer includes the name of a variable inside the program, enclose the name of the variable in single quotation marks. Do not include any information other than the return value, such as the process flow, in your answer.
  % 説明させたいメソッドを与える
```
```