Vol.2022-SE-210 No.13
2022/3/11

オブジェクト指向プログラムの要素関係とメソッドの構文木を併用したクラス名推薦

萬塲 大登[1,a)]　早瀬 康裕[1,b)]　天笠 俊之[1,c)]

概要：ソフトウェア開発におけるプログラム理解において，識別子名はドメイン知識のための重要な情報源となり得る．このためプログラムの識別子には適切な名前を与える必要があるが，適切な命名にはソフトウェア開発への熟練を要する．このような命名作業を支援するために，開発者に識別子名の候補を推薦する手法が提案されている．そのうち，メソッドの構文的な性質とメソッドの呼び出し関係を同時に考慮してメソッド名を推薦した手法では，構文的な性質のみを考慮して推薦した手法と比べ，推薦精度が向上した．このことから，プログラムを複数の観点で考慮することが推薦精度の向上に有効である可能性があると考えられる．そこで本研究では，開発者にクラスの名前を推薦するにあたり，構文的な性質とプログラム要素同士の関係を同時に考慮する手法を提案する．我々の研究グループは以前に，オブジェクト指向プログラムの要素関係を学習してクラス名を推薦する，グラフニューラルネットワークを用いた手法を提案しており，提案手法ではこれに構文的な性質を考慮する機構を追加する．すなわち，プログラムの要素関係とクラスの近傍のメソッドの抽象構文木を同時に考慮して，クラス名を推薦するグラフニューラルネットワークを構成する．抽象構文木を考慮する機構を追加する前のベースライン手法と，機構を追加した提案手法のそれぞれでクラス名の推薦精度を調べる実験を行い，クラス名推薦においてプログラム要素の関係と構文的な性質を同時に考慮することの有効性を考察する．

1. はじめに

プログラムのソースコードの約 7 割を占める [1] 識別子名は，開発者がプログラム理解を行うにあたり，アプリケーションのドメイン知識の重要な情報源となり得る [2] ため，識別子に端的な名前が付いていることが望ましい．熟練した開発者であっても作業時間の約 6 割をプログラム理解のための活動（エラーを解消するための検索エンジン利用など）に割いているといわれ [3]，この負担を軽減するために識別子により適切な名前を与えることが有意義である．

理解しやすい識別子名は開発者のプログラム理解を促進するが，命名作業は開発者にとって困難な作業となる場合がある．適切な命名には，プロジェクトの中でその識別子がどのような役割であるか [4]，プロジェクトのほかの箇所ではどのような名前が使われているか [1] といった事項を理解している必要があるためである．

命名支援を目的とした識別子名推薦において，米内ら [4] は，メソッドの構文構造と呼び出し関係の両方を推薦に利用することが，どちらかを単独で利用するよりも推薦精度の面から有効であることを示した．構文構造を考慮するメソッド名推薦手法 [5] に対し，メソッドの呼び出し関係も同時に考慮する拡張を加えることで，推薦精度を高めた．

これに着想を得た本研究では，プログラムを複数の観点で考慮して，高い精度でクラス名を推薦できる手法の提案を行う．クラス名推薦のために考慮する観点を，米内らの手法と対応させる形で，構文構造と，クラス・メソッド・フィールド同士の関係とする．

本稿の以降の構成を述べる．2 節では，本研究に関連する識別子名推薦の既存手法について説明する．3 節で提案手法を述べ， その評価を行うための実験について， 4 節で結果と考察を述べる．最後に 5 節で本稿のまとめを述べる．

2. 関連研究

識別子への命名を支援することを目的として，開発者に識別子名を推薦する手法が提案されている．以下に紹介する既存手法では，推薦される識別子の質を測る尺度として，既存の Java プロジェクト上の識別子と同じ名前を推薦するタスクを行ったときの精度を用いている．ある推薦手法での精度が高いほど，その手法が推薦する名前の質が高く，

[1] 筑波大学
University of Tsukuba
[a)] mamba@kde.cs.tsukuba.ac.jp
[b)] hayase@cs.tsukuba.ac.jp
[c)] amagasa@cs.tsukuba.ac.jp

```
int distance(int a, int b) {
  return Math.abs(a - b);
}
```

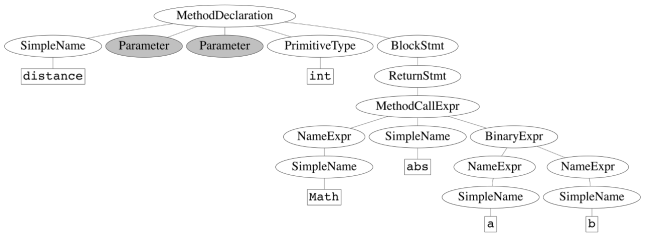

図 1 あるプログラム片とそれに対する AST の例．背景がグレーのノードについては，その子ノードを省略している．

その手法を利用すると命名作業にかかる負担が削減される可能性が高くなると考えられる．

メソッドの構文的な情報をメソッド名推薦に利用する手法として，code2vec[5], code2seq[6] がある．これらの手法では，メソッドの抽象構文木（AST，図 1）を考慮してメソッド名を推薦するために，メソッドを「AST パス」の集合として表現し，その集合に対する潜在ベクトルを計算し，計算した潜在ベクトルに基づいて推薦する名前を出力する．「AST パス」は，AST の任意の 2 つのリーフノード間のパス上に存在するノードを列挙したものである．AST パスのリーフノードには，ソースコード上に記述された，対応するトークンの情報が紐づいている．code2vec と code2seq とでは，AST パスの集合から潜在ベクトルを計算する方法が異なっており，メソッド名の推薦精度の観点から code2seq が有利である．code2vec では，AST パス全体に対して埋め込みベクトルを与えるが，code2seq では AST パスを構成する各ノードに対して埋め込みベクトルを与える．このため，code2seq では code2vec よりも，一部だけが異なる AST パス同士に近い潜在ベクトルを与えやすく，構文的な性質が異なるが意味が近いコード片同士を似たものとして考慮できる．

AST を利用したメソッド名推薦に対し，米内らは，呼び出し関係も併用して考慮することで， AST のみを利用するのに比べて高い精度で名前を推薦可能な手法 [4] を提案した．米内らは code2vec とメソッド名の推薦精度を比較する実験を行い，code2vec と比べて推薦精度が向上したことを報告している．この結果は，メソッド名推薦において，構文木のみを利用するよりも呼び出し関係を同時に考慮することが，精度向上に有効であることを示している．

以前に我々の研究グループは，プログラム要素の関係をグラフニューラルネットワーク (GNN) で考慮することでクラス名を推薦する手法 [7] を提案している．この手法は，クラス，メソッド，フィールド同士の継承，所有などを表すグラフを入力し，グラフ構造を学習に利用可能な GNN によって，推薦するクラスの名前の候補を出力する．推薦対象クラスの近傍ノードを潜在ベクトルにエンコードする部分，潜在ベクトルをノード間で伝播する部分，伝播した潜在ベクトルをもとに推薦するクラス名の候補を出力する 3 部分で構成される．

3. 提案手法

本節では，開発者に質の良いクラス名の候補を与えることを目的とした，プログラムの要素関係と抽象構文木 (AST) を併用してクラス名を推薦する手法について述べる．推薦システムは，図 2 に示すように，入力で受け取ったソースコード群から構築したプログラム要素の関係グラフと，メソッドのノードに対応する AST の 2 種類を利用して，開発者が名前の推薦を受けたいと考えているクラス（以降「推薦対象クラス」と呼ぶ）の名前を出力する．

要素関係グラフと AST の併用によって，クラスの機能をより正確に捉えた推薦が可能になると期待される．要素関係グラフを利用することで，推薦対象クラスの近傍のプログラム要素の性質を考慮できる．そのような性質は，AST を利用することでより正確に捉えられるようになると考えられる．米内らのメソッド名推薦手法 [4] では，メソッドの呼び出し関係と AST の併用によって推薦精度を向上させている．よって，提案手法のクラス名推薦では，要素関係グラフと AST の併用が推薦精度の向上に寄与する可能性がある．

クラス名への推論には，グラフニューラルネットワーク (GNN) を用いる．この GNN のうち，要素関係からクラス名の単語列を推薦する部分を，我々の研究グループが提案したクラス名推薦手法 [7] と同様に構成し，メソッドの AST を考慮する機構を追加する．すなわち，推薦対象クラス近傍の各ノードに対応する潜在ベクトルを推薦対象クラスのノードへ伝播させ，伝播で更新された潜在ベクトルからクラス名の単語列が出力される GNN とする．また，メソッドのノードには code2seq [6] で計算される潜在ベクトルを割り当てる．このことを図 3 に示す．

GNN による潜在ベクトルの伝播により，推薦対象クラスの近傍ノードに対し，その周辺ノードの性質を併せ持つ潜在ベクトルが得られると考えられる．この潜在ベクトルには，近傍メソッドの AST の情報も含むため，そのようなメソッドの構文木を考慮できる．例えば ‘`BufferedReader`’ クラスから ‘`readLine()`’ メソッドへの所有関係のエッジがあれば， ‘`BufferedReader`’ クラスに対応する潜在ベクトルは， ‘`readLine()`’ メソッドの構文木の情報を含む．

以下本節では，提案手法のニューラルネットワークの具体的な構成を述べる．

3.1 GNN による要素関係の考慮と推薦クラス名の出力

プログラム要素の関係を考慮し，推薦するクラス名の候補を出力する部分の大まかな構成は，我々の研究グループ

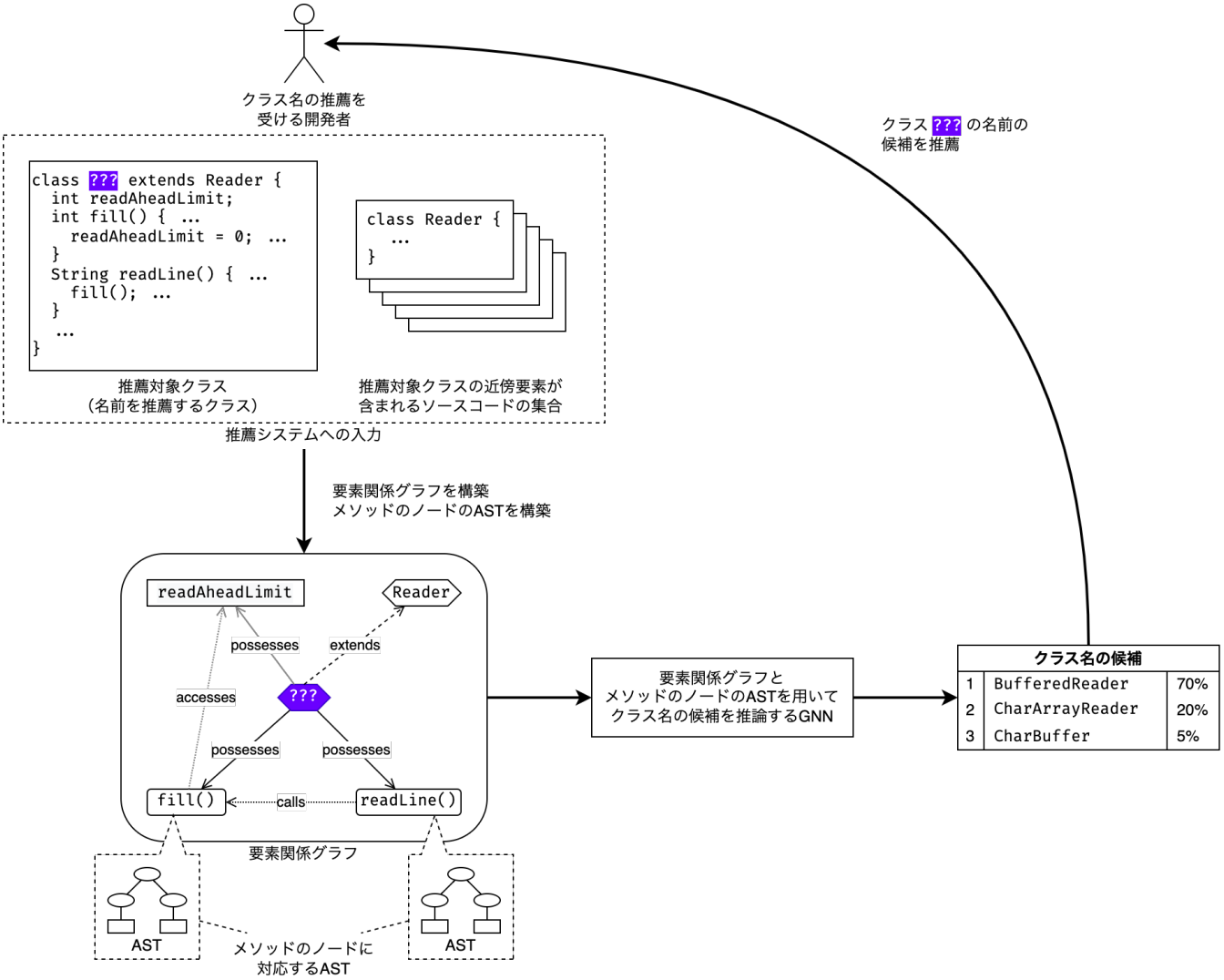

図 2 提案手法の構成

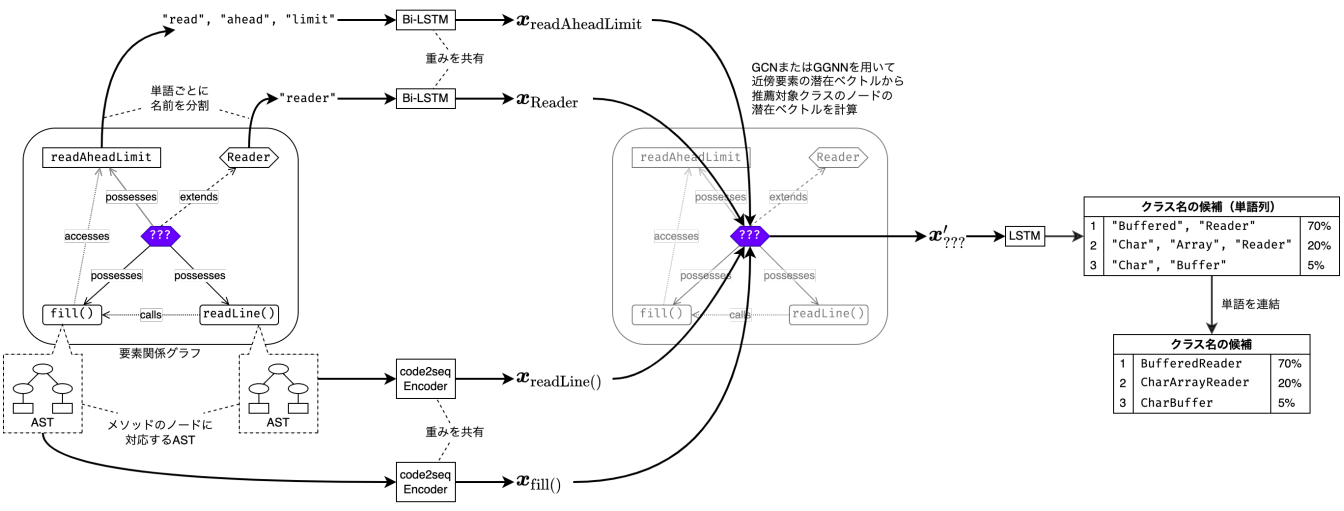

図 3 ニューラルネットワークの構成

が以前に提案した手法 [7] に従う．すなわち，プログラム要素がもつ潜在ベクトルをグラフニューラルネットワークで伝播させ，クラス名の候補を単語列として出力するようにする．

入力する要素関係グラフの構成は， [7], [8] での構成と同様とする．すなわち，プログラム中のクラス，メソッド，フィールドをノードとし，次の 7 種類の関係をエッジとしたグラフを入力する．

- クラス同士の継承関係.
- クラスがフィールドを所有する関係.
- クラスがメソッドを所有する関係.
- メソッド同士の呼び出し関係.
- メソッドがフィールドへアクセスする関係.
- メソッドとその戻り値の型を表現するクラスとの関係.
- フィールドとその型を表現するクラスとの関係.

潜在ベクトルを推薦対象クラスのノードへ伝播するグラフニューラルネットワークとして Gated Graph Neural Network (GGNN) [9] または Graph Convolutional Network (GCN) [10] を利用し，伝播で得られた新たな潜在ベクトルを長・短期記憶 (LSTM) [11] へ入力し，推薦するクラス名の候補を単語列として出力する．

3.2 プログラム要素の性質のエンコード

GNN で伝播させる潜在ベクトルにプログラム要素の情報を含めるために，ノードの種類に応じて潜在ベクトルへエンコードする．メソッドに対応するノードについては AST を利用し，その他のノードにはそのプログラム要素の名前の単語列を利用する．

3.2.1 構文木によるエンコード

code2seq [6] のうち，AST パスから潜在ベクトルを計算する部分を利用して，メソッドに対応するノードをエンコードする．

つまり，メソッドの AST パスの集合の各要素について，AST ノードの系列データおよびリーフノードのトークンから潜在ベクトルを計算し，全要素について潜在ベクトルの平均をとったものをそのメソッドの潜在ベクトルとする．

3.2.2 プログラム要素の名前によるエンコード

3.2.1 節で述べた構文木によるエンコードを行わない要素では，その要素の名前を利用して潜在ベクトルにエンコードする．そのような要素とは，クラスやフィールドに加え，入力データの都合でメソッドボディを解決することのできないメソッドを指す．

プログラム要素の名前を `camelCase` や `snake_case` によって単語ごとに分割したもの（例えば，*LineNumberReader* という名前は *Line*, *Number*, *Reader* という 3 単語の列になる）を bi-directional LSTM へ入力したときの隠れ状態を，そのプログラム要素のグラフノードに対応する潜在ベクトルとする．

3.3 ニューラルネットワークの訓練

この GNN の訓練では，出力する各単語についての確率分布と ground truth における単語との交差エントロピー損失を最小化するための教師付き学習を行う．損失の最適化アルゴリズムとして， Adam [12] を用いる．

ここで，推薦対象クラスの ground truth となる名前が，要素関係グラフのほかの要素の名前に含まれていたり，メソッド中のトークンに含まれていたりする場合，それらをマスキングする．例えば，推薦対象クラスの ground truth となる名前が ‘`ArrayList`’ であるとき，‘`getArrayListItem`’ という 4 単語のトークンは *get*, ‘`<SLOT>`’, *Item* の 3 単語とする．

4. 評価実験の計画

クラス名の推薦にあたって要素関係とメソッドの AST を併用することの有効性を評価するため，要素関係のみを利用した場合と提案手法とで推薦精度を比較する実験を行う．実験のデータセットとして既存のオープンソースプロジェクトのソースコード群を利用し，ソースコード中のクラスの名前を ground truth とする．データセットに含まれる大量のクラスに対しそれぞれの方法でクラス名を推薦し，ground truth に対する推薦精度を計測する．

本節の以降の部分では，この実験の計画について述べる．

4.1 比較手法

提案手法との対比のために，要素関係のみを利用してクラス名を推薦する手法と推薦精度を比較する．すなわち，提案手法ではクラス名推薦のためにメソッドの AST を考慮できる一方，この比較手法では AST を考慮できない．これは我々が以前に提案したクラス名推薦手法 [7] と同様の構成である．

3.2 節で述べたプログラム要素から潜在ベクトルへのエンコードについて，比較手法では提案手法と異なり，要素の名前によってのみ行う．提案手法では，要素関係グラフでメソッドに対応するノードに対しては AST から潜在ベクトルを計算し（3.2.1 節），その他のノードに対しては要素の名前によって潜在ベクトルを計算する（3.2.2 節）が，比較手法では，プログラム要素の種類に関わらず後者の方法で潜在ベクトルを計算する．

4.2 データセット

この実験のためのデータセットとして，質が良いと考えられるオープンソースプロジェクトのソースコード群を利用する．データセットに含まれる各クラスについて，ソースコード中に実際に記述されているクラス名を，そのクラスの名前を推薦するときの ground truth とする．

実験に利用可能なデータセットには，Alon らが code2seq [6] の評価のために用いたものが挙げられる．これは，ソー

スコードのホスティングサービスの GitHub において，Java プロジェクトのうちスター数が上位 1000 位以内のプロジェクトのソースコードを収集して構築されたデータセットである．GitHub でスターが多く付くプロジェクトは，多くの利用者を持ち，また，多くの開発者が長期間にわたって活発にメンテナンスを行っていると考えられる．このようなプロジェクトではソースコードにある程度の質が担保されており，ソースコード中の識別子名も同様であると考えられるため，本実験でも同じデータセットを利用する．Alon らはこのデータセット中の 1000 プロジェクトを，800 : 100 : 100 の比率で訓練用，検証用，テスト用に分割している．本実験でも各プロジェクトを Alon らと同じ用途で用いる．

4.3 実験方法

提案手法と比較手法のそれぞれで，訓練済みの推薦モデルを利用して，テスト用データに含まれるクラスの名前をどの程度正しく推薦できるかを計測する．

推薦精度の計測に先立って，推薦モデルの訓練とハイパーパラメータのチューニングを行う．訓練用データを用いて 3.3 節の訓練を行い，検証用データによる F 値がなるべく大きくなるようにする．また，Optuna [13] を利用し，検証用データによる F 値がなるべく大きくなるハイパーパラメータを探索する．

4.3.1 評価指標

評価指標には，[7] と同様に，テスト用データに含まれるクラスの名前を推薦したときの ground truth に対する適合率・再現率・ F 値と，「部分一致 (k)」・「完全一致 (k)」を用いる．なお，大文字・小文字の区別は無視する．これらの評価指標の値が比較手法より提案手法で大きければ，クラス名推薦で構文木を併用することが有効であるといえる．

適合率・再現率・F 値は，推薦結果の最上位のクラス名および ground truth に含まれる単語について，true positive, false positive, false negative を数え上げて計算する．例えば，ground truth が *LineNumberReader* であるクラスと *LinkedList* であるクラスの各名前を推薦した結果が *Reader*, *LinkedArrayList* であった場合，true positive の単語は *Reader*, *Linked*, *List* の 3 つ，false positive の単語は *Array* の 1 つ，false negative の単語は *Line*, *Number* の 2 つである．よって，適合率は $3/4 = 0.75$，再現率は $3/5 = 0.6$，F 値は約 0.67 となる．

「部分一致 (k)」と「完全一致 (k)」は，推薦結果上位 k 位以内に ground truth に部分一致または完全一致するクラス名を推薦できた割合である．k の値として 1 および 10 を使用する．部分一致 k については，ground truth に含まれるいずれかの単語が推薦結果に含まれていれば一致とみなす．

全体一致 (k) 以外の指標は，クラス名を構成する単語がどの程度適切に出現するかを示す．識別子名推薦の実際の利用場面では，推薦されたクラス名の候補をもとに，開発者自身がその候補よりも適切な名前を検討する場合が考えられる．この場合，推薦されたクラス名が ground truth に完全に一致しなくとも，一部の単語が一致していればその候補は有効に利用される．適合率・再現率・F 値・部分一致 (k) の値が大きければ，提案手法はこのような場面で有効に作用するといえる．

4.4 実装

実装言語に Python 3.9.9 を使用する．また，機械学習ライブラリとして Pytorch 1.10.1 [14] を使用し，グラフニューラルネットワークの学習のために Deep Graph Library 0.7.2 [15] を使用し，ハイパーパラメータの最適化のために Optuna 2.10.0 [13] を使用する．

4.5 実験の進捗状況

これまでに予備実験として，4.2 節で述べたものより小規模なデータセットを用いて提案手法の精度を調べる実験を行っている．このデータセットは，4.2 節で述べたものと同じく Alon らが code2seq [6] の評価で用いたもので，訓練用の 9 プロジェクト，検証用の 1 プロジェクト，テスト用の 1 プロジェクトの計 11 プロジェクトのソースコードで構成されている．

このデータセットの検証用データでの F 値は 0.0418 であり，低い水準であった．今後，4.2 節で示した大規模なデータセットで実験を行い，提案手法と比較手法で推薦精度を比較する予定である．

5. 結論

本研究では，開発者によるクラスへの命名がときに困難な作業となる問題に対し，プログラムの要素関係とメソッドの構文木を利用してクラス名を推薦する手法を提案した．この手法では，プログラム要素の関係を表すグラフと抽象構文木 (AST) の 2 種類を入力にとり，グラフニューラルネットワークで近傍の要素の性質を考慮しながら，クラス名の候補を出力する．

今後は提案手法の有効性を評価するため，要素関係のみを利用する場合と推薦精度を比較する実験を行う予定である．既存のオープンソースプロジェクトに含まれる大量のクラスに対し，ソースコード上に記述されたクラス名を正しく推薦できる割合を計測する．

参考文献

[1] Deissenboeck, F. and Pizka, M.: Concise and Consistent Naming, *Software Quality Journal*, Vol. 14, No. 3, p. 261–282 (online), DOI: 10.1007/s11219-006-9219-1

(2006).
[2] Lawrie, D., Morrell, C., Feild, H. and Binkley, D.: What's in a Name? A Study of Identifiers, *14th IEEE International Conference on Program Comprehension (ICPC'06)*, pp. 3–12 (online), DOI: 10.1109/ICPC.2006.51 (2006).
[3] Xia, X., Bao, L., Lo, D., Xing, Z., Hassan, A. E. and Li, S.: Measuring Program Comprehension: A Large-Scale Field Study with Professionals, *IEEE Transactions on Software Engineering*, Vol. 44, No. 10, pp. 951–976 (online), DOI: 10.1109/TSE.2017.2734091 (2018).
[4] 米内裕史，早瀬康裕，北川博之ほか：ソースコード構文木とコールグラフの統合的な埋め込みに基づくメソッド名の推定，研究報告ソフトウェア工学 (SE)， Vol. 2020, No. 10, pp. 1–8 (2020).
[5] Alon, U., Zilberstein, M., Levy, O. and Yahav, E.: Code2Vec: Learning Distributed Representations of Code, *Proc. ACM Program. Lang.*, Vol. 3, No. POPL, pp. 40:1–40:29 (online), DOI: 10.1145/3290353 (2019).
[6] Alon, U., Brody, S., Levy, O. and Yahav, E.: code2seq: Generating Sequences from Structured Representations of Code, *International Conference on Learning Representations*, (online), available from ⟨https://openreview.net/forum?id=H1gKYo09tX⟩ (2019).
[7] 萬場大登，早瀬康裕，天笠俊之，北川博之：オブジェクト指向プログラムの要素関係グラフを用いたクラス名の end-to-end 学習，第 83 回全国大会講演論文集，Vol. 2021, No. 1, pp. 277–278 (2021).
[8] Kurimoto, S., Hayase, Y., Yonai, H., Ito, H. and Kitagawa, H.: Class Name Recommendation Based on Graph Embedding of Program Elements, *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, pp. 498–505 (2019).
[9] Li, Y., Zemel, R., Brockschmidt, M. and Tarlow, D.: Gated Graph Sequence Neural Networks, *Proceedings of ICLR'16*, (online), available from ⟨https://www.microsoft.com/en-us/research/publication/gated-graph-sequence-neural-networks/⟩ (2016).
[10] Kipf, T. N. and Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks (2017).
[11] Hochreiter, S. and Schmidhuber, J.: Long short-term memory, *Neural computation*, Vol. 9, No. 8, pp. 1735–1780 (1997).
[12] Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Bengio, Y. and LeCun, Y., eds.), (online), available from ⟨http://arxiv.org/abs/1412.6980⟩ (2015).
[13] Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M.: Optuna: A Next-generation Hyperparameter Optimization Framework, *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019).
[14] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, *Advances in Neural Information Processing Systems 32* (Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. and Garnett, R., eds.), Curran Associates, Inc., pp. 8024–8035 (online), available from ⟨http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf⟩ (2019).
[15] Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J. and Zhang, Z.: Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks, *arXiv preprint arXiv:1909.01315* (2019).