DEIM Forum 2019 G1-4

コールグラフの分散表現を利用したメソッド名予測手法の定量的評価

米内　裕史† 　早瀬　康裕†† 　北川　博之†††

† 筑波大学 システム情報工学研究科　〒 305-8573　茨城県つくば市天王台 1-1-1
†† 筑波大学 システム情報系　〒 305-8573 茨城県つくば市天王台 1-1-1
††† 筑波大学 計算機科学研究センター　〒 305-8577 茨城県つくば市天王台 1-1-1
E-mail: †yonai@kde.cs.tsukuba.ac.jp, ††{hayase,kitagawa}@cs.tsukuba.ac.jp

あらまし　プログラムのソースコード中の識別子がその役割や動作を示した名前を持つことは，そのソースコードの可読性の重要な要件である．しかし，識別子に対して理解の手がかりとなる適切な名前を付けることは，そのソフトウェアが対象としているドメインの知識を必要とする，困難な作業である．そこで我々はメソッドの内部の情報からメソッド名として適切な名前を予測する手法を提案した．この手法は，メソッドとその内部で使用されているメソッド集合との間の呼び出し関係のグラフに基づきメソッド名に対する分散表現を生成し，その近傍を探索することでメソッド名の候補を決定するものである．本稿では，この手法とメソッド名の推薦を行うような既存手法を多面的な観点から評価・比較し，考察を行う，具体的には，どのような単語をもつメソッドに対して有効なのか，どういう役割をもつメソッドに対して有効なのか，などの観点から評価する．

キーワード　識別子名, メソッド名, コールグラフ, 分散表現

1 序　　論

ソフトウェア開発において，開発対象のプログラムを理解する作業は重要である．以前主流であった，ウォーターフォールモデルと呼ばれる開発工程全体を作業工程で分割しトップダウンに進める開発手法では，全作業工程のうちリリース後のソフトウェア保守にかかるコストが約 7 割を占める [1]．ソフトウェア保守の作業では，保守対象となるプログラムの修正箇所をソースコードから読み解く必要があり，この作業にかかるコストは保守作業全体のうち大部分を占める [2]．近年主流となりつつある，アジャイル開発と呼ばれる開発対象を多数の小さな機能に分割し段階的に変更を加えながらリリースを繰り返す開発手法では，円滑にリリースを続けるためには各機能がどのようなプログラムによって実装されているのかを理解することが重要であるので，プログラムを理解する作業が占めるウェイトはますます大きくなっている．

プログラムを理解する際に，保守作業者はプログラムのソースコード中に出現するクラス名，メソッド名，変数名といった識別子名を動作を理解する手がかりとして利用する [3]．そのため，開発者はプログラム中の要素に対してその動作や役割を示すような識別子名を与えることで，保守作業者がプログラムを理解するのに要するコストを減らすことができる．しかし，識別子に対してそのような適切に示すような識別子名を与えることは，そのプロジェクトに対する包括的な理解や，識別子の命名に関する知識を必要とする難易度の高い作業である．

そこで，開発者が識別子を命名するタスクを支援することを目的として，ソースコードの情報から識別子名を評価，推薦する手法が提案されている．柏原ら [4, 5] の手法は，メソッドの定義の情報をメソッド名の推薦に利用できることを示したものであるが，メソッド名を構成する単語群のうち，名詞部分の予測に関しては動詞部分の予測と比較して精度が低いという課題がある．Allamanis ら [6] の手法は，メソッド名やその構成単語に関する分散表現がメソッド名の推薦に利用できることを示したものであるが，分散表現を生成するためにプログラム中でメソッドが呼び出されている文脈を必要とするので，メソッドを定義した時点では名前の推薦を行うことができないという問題がある．

我々は，メソッドの予測を行う手法を別に提案した [7]（以下，提案手法と呼称）．提案手法は，メソッドを定義した時点でメソッド名の予測を可能にすること，また，予測精度，特に名詞部分に関する予測精度を既存手法より向上させることを目的とするものである．この手法では，予測の根拠として，メソッドの名前と内部で呼び出しているメソッド集合の名前との間の関連性を利用した．また，既存のソースコードよりそのような関係を抽出し，抽出した関係を利用して各メソッド名に対してメソッド名の意味関係の情報を保持するような分散表現を生成した．メソッド名の予測は，これらの分散表現の位置関係を利用して行う．

我々は [7] において，提案手法によるメソッド名の推薦精度を評価する実験を行った．その結果は，メソッドの呼び出し関係に基いて生成した分散表現が，メソッド名の予測に活用できることを示唆するものであった．また，提案手法が名詞部分に関しても動詞部分と同様の精度で予測できることを示唆するものであった．

本稿では，提案手法とメソッド名の推薦を行うような既存手法をより多面的な観点から評価・比較し，考察を行う，具体的には，どのような単語をもつメソッドに対して有効なのか，どういう役割をもつメソッドに対して有効なのか，などの観点から評価を行い，既存手法との比較を行う．

2 節で本研究における基本事項について説明する．次に 3 節で提案手法について概要を述べる．4 節で本稿において行った評価実験について述べる．5 節で実験の結果の考察を述べる．最後に 6 節で本研究についてまとめ，今後の方針について述べる．

2 基本事項

本節では本研究の基本事項について述べる．2.1 小節では提案手法や一部の既存手法で利用されている単語の分散表現について述べる．2.2 小節ではメソッド名の推薦を目的とした既存手法について述べる．

2.1 単語の分散表現

本節では，本研究や一部の先行研究において識別子名の評価に利用した単語の分散表現について説明する．単語の分散表現は，自然言語の単語を一定数の次元の実数値のベクトルで表す手法 [8] であり，これによって単語を豊富な情報量を保持した上で定量的に表現することができる．例えば，ベクトルの位置関係によって，単語間の意味の類似関係を表現することができる．単語の分散表現は，自然言語処理の分野において活発に利用されており，自然言語における単語に関する分散表現を得る手法が提案されている．Mikolov ら [9–11] は分布仮説 [12] に基づき，文章中に出現する各単語の前後で出現する単語によって，注目している単語の分散表現を得る手法を提案した．

2.2 既存手法

開発者が識別子を命名するタスクを支援するため，ソースコードの情報から識別子名を評価，推薦する手法が提案されている．Høst らは，メソッド名を構成する単語のうち動詞部分に注目し，メソッドの処理内容とメソッド名の動詞の関係を調査した．また，そのような関係を収録した辞書を作成する手法を提案した [13]．さらに，メソッド名とメソッドの処理内容の関係を抽出し，抽出したルールに沿わないような不適切なメソッド命名を判別し，より適切なメソッド名を提案する手法を提案した [14]．柏原ら [4,5] は，メソッドの名前とその内部で使用している変数やメソッド，引数の型などの間に関連があると考え，その間の相関ルール [15] を抽出し，既に定義が存在するようなメソッドに対して，より良いメソッド名を推薦する手法を提案した．この手法はメソッドの定義の情報をメソッド名の推薦に利用できることを示したものであるが，メソッド名を構成する単語群のうち，名詞部分の予測に関しては動詞部分の予測と比較して精度が低いという課題がある．Allamanis らは，既存のソースコード群からメソッドが呼びされている箇所の前後に出現するプログラム要素から各メソッド名に関する分散表現を生成し，それを利用してメソッド名の予測を行う手法を提案した．また，識別子名を構成要素である単語に分割し，単語それぞれに関して分散表現を生成し，メソッド名を単語の組み合わせと見なして予測を行うことで，既存のソースコード群に含まれていないような新たなメソッド名を提案する手法を提案した [6]．この手法はメソッド名やその構成単語に関する分散表現がメソッド名の推薦に利用できることを示したものであるが，分散表現を生成するためにプログラム中でメソッドが呼び出されている文脈を必要とするため，メソッドを定義した時点では名前の推薦を行うことができないという問題がある．また，メソッドの定義から，そのメソッドの数単語の要約を生成する手法を提案した [16]．この手法では，メソッドの定義に該当するソースコードをトークン列に変換したものを入力とし，そのメソッドの動作に関する数単語の要約を出力とする深層学習モデルを用いる．このモデルの特徴は，トークン列のうちどのトークンを重視して出力の決定に用いるかという，トークン列に対する重み付けを行うアテンション機構をモデルに導入したことである．Allamanis らは論文の実験において，このモデルの出力をメソッド名と見なした場合の正しいメソッド名を予測する精度を検証した．

3 提案手法

本節では提案手法である，メソッド間の呼び出し関係に基づいてメソッド名の分散表現を生成し，メソッド名の予測を行う手法について説明する．既存手法の課題を克服するため，提案手法ではメソッドの定義内の情報を用いてメソッド名に関する分散表現を獲得したのち，推薦対象メソッドを表す分散表現を生成しその近傍に分散表現が存在するメソッド名を探索することで推薦を行う．分散表現を利用することでその位置関係から単語間の意味の関連を表現できるため，柏原らの手法における名詞の多様性に起因する名詞部分の予測精度が低い課題をを克服することが期待される．また，メソッドの使用されている文脈に依存せず，定義の内部の情報のみを用いて分散表現を生成するため，Allamanis らの手法におけるメソッドを定義した時点でメソッド名の予測ができない課題を克服することが期待される．

図 1 に提案手法全体の概要図を示す．提案手法は大きく分けて分散表現を生成する前処理の部分とメソッド名の予測を行う予測処理の部分で構成される．

前処理では，まずメソッド間の呼び出し関係を表現したグラフであるコールグラフを既存のソースコード集合より生成する．生成したコールグラフの構造に基づいて，各メソッドに対してどのような分散表現を期待するかを決定する (3.1 小節)．期待する分散表現に可能な限り近い分散表現を得るために，各メソッドの分散表現に関する損失関数を設定した．この損失関数は，各メソッドの分散表現がメソッド間の呼び出し関係を表現できていればいるほど，値が小さくなるようなものである．この損失関数の値を小さくするように分散表現を最適化することで各メソッドに関する分散表現を獲得する (3.2 小節)．

予測処理では，定義が存在するがメソッド名が不明であるようなメソッドが与えられた時に，そのメソッドの内部で呼び出しているメソッド集合より，そのメソッドを表す分散表現を生成し，ベクトル空間上でその分散表現の近傍を探索する．この時近傍に存在する既存のメソッド名に関する分散表現をメソッド名の候補とし，予測を行う (3.3 小節)．

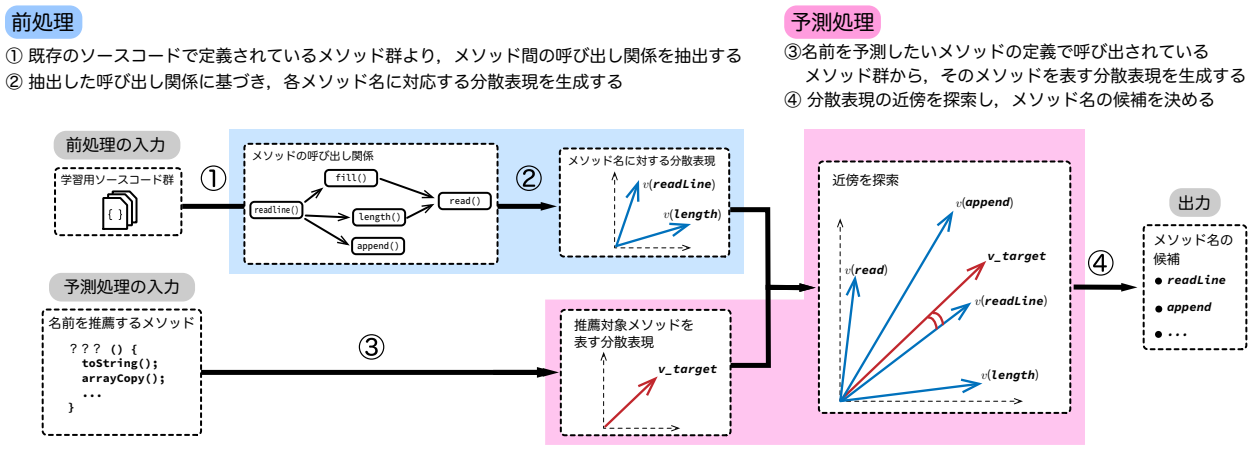

図 1　提案手法の概要図

3.1　コールグラフの生成およびそれに基づく分散表現の決定

本小節では既存のソースコード集合からメソッドのコールグラフを生成し，その構造によって各メソッドの分散表現を定める流れについて述べる．コールグラフは，ノードがメソッドを表し，呼び出し元のメソッドのノードから呼び出し先のメソッドのノードに有向エッジが張られるようなグラフである．このようなコールグラフを，既存のソースコードから抽出したメソッドの呼び出し関係より生成する．

生成したコールグラフの各ノードが示すメソッド名に対応する分散表現の値が，そのノードを始点とするエッジの先のノードが示すメソッド集合，即ち定義の内部で呼び出しているメソッド集合の分散表現の平均の値に近づくように，各メソッドに対応する分散表現の値を定める．図 2 に概要図を示す．これにより，メソッドの内部の動作，即ち呼び出しているメソッドによって呼び出し元のメソッドの分散表現を決定することを表現する．また，内部で呼び出しているメソッド群の分散表現の平均を取ることにより，呼び出し元のメソッドの分散表現と内部で呼び出しているメソッドの分散表現の間で意味的な関連を表現できることが期待される．

このとき，コールグラフを正確に生成するためには，各メソッドが定義の内部で呼び出しているメソッドがどのパッケージ，クラスに属するものであるのか，また各メソッドはどのパッケージ，クラスに属するメソッドの内部で呼び出されているのかといった，メソッドの名前解決を正確に行う必要がある．しかし，コールグラフの構築にあたりいくつかの場合でソースコード中のメソッドの名前解決ができない場合があり，完全なコールグラフを生成するのは困難である．なぜならば，メソッドはクラスやパッケージが異なれば 1 つのメソッド名で複数の定義が存在し得るが，名前解決ができない場合，メソッドのボディで呼び出している他のメソッドがそのメソッド名に対応する複数の定義のうち，どの定義を指しているものなのか区別できないからである．対策として，同名のメソッド定義が複数あった場合に，コールグラフ上でそれらをクラスやパッケージにより区別せずメソッド名のみで識別し，ひとつのノードに統合するという処理を行なう．このとき，統合されるそれぞれの

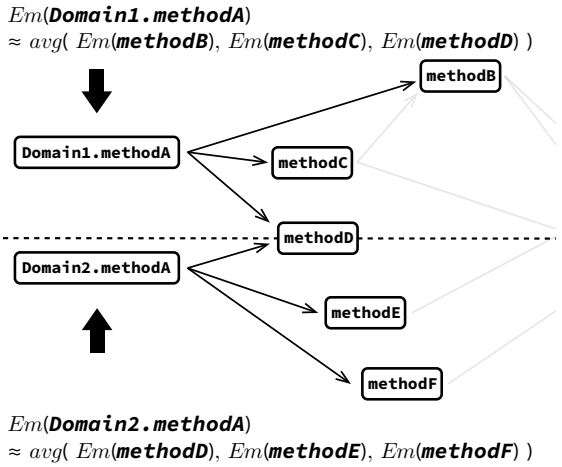

図 2　コールグラフより分散表現の値を決定する例

メソッドの定義の内部で呼び出しているメソッド群はすべて統合後のひとつのメソッドが呼び出しているものとする．

3.2　損失関数およびその最適化

本小節では，前小節で述べたような分散表現を獲得するために定めた，各メソッドの分散表現に関する損失関数について述べる．

獲得する分散表現に関する損失関数 E を，以下のように定義した．

$$E = \alpha \sum_{k=1}^{n} |Em(m_k) - avg(Em(t_{k1}) + Em(t_{k2}) + \ldots)|^2 + (1-\alpha) \sum_{k=1}^{n} |1 - Em(m_k)|^2$$

関数 E において，n はコールグラフ上のメソッドの数，$Em(m_k)$ は k 番目のメソッドに関する分散表現，$Em(t_{ki})$ は k 番目のメソッドの定義の内部において i 番目に呼び出されているメソッドの分散表現を示す．この関数は 2 つの項で構成され，第 1 項は各メソッド名に対応する分散表現が，内部で呼び出しているメソッド群の分散表現の平均に近いほど値が小さく

なる．第 2 項はは各メソッドのノルムが 1 に近づくほど値が小さくなるもので，分散表現が零ベクトルに収束することを防ぐためのものである．

最適化のアルゴリズムには確率的勾配降下法を使用する．今回の問題では，最適化中の各ステップでコールグラフ上のメソッドをランダムにいくつか選択し，そのメソッドに関する損失関数 E の勾配を算出し，分散表現の値の更新を行なう．

また，分散表現群がひとつのベクトルに収束することを防ぐため，確率的勾配降下法において各ステップで，注目しているメソッドと呼び出し関係の無いメソッドをいくつか選択し，それらのメソッドの分散表現から離れるような勾配の項を追加する．i 番目のメソッドに注目しているとき，呼び出し関係の無いメソッドをいくつか選択しそのインデックス集合を $\boldsymbol{N}$ とすると呼び出し関係の無いメソッドの分散表現から離れるような項 g_i は

$$g_i = \sum_{k \in \boldsymbol{N}} -(o_{Em(m_k)} - Em(m_i))$$

で表される．

以上より，損失関数の最適化では，確率的勾配降下法の各ステップで損失関数 E に基づく勾配と，g_i に基づく勾配を考慮して分散表現の値の更新を行なう．

3.3 分散表現に基づいたメソッド名予測

メソッド名の予測は，既に定義が存在するがメソッド名が不明もしくは改名するような場面を想定して行う．まずメソッドの定義の内部で呼び出しているメソッド集合を抽出する．抽出したメソッド集合に関する分散表現の平均の値 $v\_avg\_callees$ を計算する．この $v\_avg\_callees$ が呼び出し元のメソッド名に期待する分散表現の値であるので，ベクトル空間上で $v\_avg\_callees$ に近い分散表現をもつメソッド名を呼び出し元メソッドのメソッド名の候補とする．近さの判定は $v\_avg\_callees$ と各分散表現とのコサイン類似度を計算することによって行い，コサイン類似度が高い順に確信度の高い候補とする．

4 評 価 実 験

本節では提案手法によるメソッド名の推薦精度を評価する実験を行う．実験の目的は，提案手法がメソッド名の命名を支援できそうかどうかを評価することである．

メソッド名の命名支援は，開発者に対してメソッド名全体を推薦するより，メソッド名の構成要素となり得る単語群を個別に推薦するほうが良い．これは，メソッド名はそのメソッドの動作上の特徴や性質を表現しているものであるべきだが，そのメソッドに関するどのような特徴や性質を表現するかによって，どのようなメソッド名が適切であるかは変わり得るからである．よって，メソッド名全体を推薦するより，開発者にとってメソッド名を決める上でキーワードとなり得るようなものを単語単位で推薦するほうが開発者にとって有益であると考えられる．

そのため，本実験では主に，あるメソッドに対して推薦結果として開発者に提示したメソッド名集合の中に，そのメソッドの正しいメソッド名と単語単位で一致するものが存在するかどうか，という観点で評価を行う．正しいメソッド名に含まれるような単語を推薦結果として提示できれば，それは適切なメソッドの命名を行う上で支援になると考えられる．

また，本実験では提案手法とメソッド名推薦を行うような既存手法との間の単純な精度の比較だけでなく，主に以下のような観点から評価・比較を行う．

- どのような単語をもつメソッド名に対して精度よく推薦ができるのか．
- どのような役割をもつメソッド名に対して精度よく推薦ができるのか．
- そもそも命名支援が必要であると考えにくいような単純なメソッド名をもつメソッドを検証用のデータから除いた場合にどの程度推薦精度に影響を及ぼすか．

比較対象には Allamanis ら [16] によって提案された手法を用いる．

モデルの学習・評価のためのデータセットには，Java のソースコード集合を用いる．提案手法のモデルの学習では，学習用ソースコード集合において定義されているメソッドからメソッド名と内部で呼び出されているメソッド群の組を抽出し，それらを用いてメソッド名に対応する分散表現の最適化を行う．評価では，評価用ソースコード集合において定義されているメソッドからメソッド名と内部で呼び出されているメソッド群の組を抽出し，内部で呼び出されているメソッド群からそのメソッド宣言に対するメソッド名の予測を行う．そして，予測によって得られるメソッド名の候補と正しいメソッド名を比較し，精度を測定する．同様に，比較手法のモデルの学習・評価では，ソースコード集合において定義されているメソッドからメソッド名とそのメソッドのトークン列の組を抽出し，学習・予測を行う．その後，提案手法による予測の精度・傾向と比較手法による予測の精度・傾向を後述の基準により比較・評価する．

メソッド名推薦の精度を測定する際は，評価用ソースコード集合において定義されている各メソッドに対して，そのメソッド名の構成単語を推薦結果として提示できるか，という指標で行う．そのため，ソースコード集合において定義されているメソッドは，メソッド名が適切に命名されていることがが求められる．本実験で用いるソースコードに関しては，後述の通りリポジトリの Watcher 数と Fork 数に基づく指標によって選定されたものであるが，これらの数値はそのリポジトリの注目度を示す指標となるものである．注目度の高いリポジトリはそれだけ多くの開発者が関わっているものであるため，メソッド名に関しても相応に議論がなされ，適切な命名が行われているものが多いと考えられる．

データセットとなる Java のソースコード集合は，分散表現を利用してメソッド名の予測を行うような先行研究である Allamanis ら [6] の手法において評価実験に使用していたデータセットを使用する．このデータセットは [17] において他者が利用できる形で公開されている．このデータセットはソフトウェア開発プロジェクトを共有するサービスである GitHub [18] に公開されている Java を用いたプロジェクト群を Allamanis らが定めた，リポジトリの Watcher 数と Fork 数に基づく指標によっ

表 1　提案手法および比較手法でデータセットから抽出したメソッド宣言の数

	提案手法	比較手法
subset 1	24,166	23,049
subset 2	25,457	23,505
subset 3	25,819	24,836
subset 4	24,168	22,048
subset 5	25,589	24,214

て選定し，ソースコードを取得したものである．なお，提案手法の評価と比較手法の評価では同じソースコード集合をデータセットとして使用しているが，ソースコードからメソッド宣言を抽出する処理が提案手法と比較手法で異なるため，評価用のメソッド集合が一致しない．評価手法のメソッド宣言を抽出するプログラムは Java のバイトコードにコンパイルされた状態で公開されていたため，どのような処理に基づいて抽出を行っていたか厳密に理解するのは困難であった．表 1 に提案手法・比較手法でデータセットから抽出したメソッド宣言の数を示す．

予測精度の評価では，以下の項目に関して評価を行う．

1. テストデータ中の全てのメソッドに関して評価する.テストデータに含まれる全メソッド宣言について，提案手法ではその内部で呼び出しているメソッド宣言から，Allamanis らの手法ではメソッド定義のソースコードのトークン列からメソッド名の予測を行う．予測結果として返されるメソッド名の候補を確信度でソートした後，その上位 10 件のメソッド名の候補集合と実際のメソッド名と比較し，予測の成否を判定する．このとき，メソッド名の動詞部分の予測の成否と名詞部分の予測の成否は別々に判定する．すなわち，メソッド名の候補 10 件の中に実際のメソッドと同じ動詞部分をもつものが存在すれば，動詞部分の予測に成功したとし，同様に実際のメソッドと同じ名詞部分をもつものが存在すれば，名詞部分の予測に成功したとする．

2. 単純なメソッド名のメソッドを除外して評価する.Java のクラス設計において，フィールドへの外部からの直接的な操作を禁止し，”getter”や”setter”と呼ばれる，フィールドにアクセスするためのメソッドを介してアクセスする設計手法が存在する．このとき，getter や setter となるメソッドの命名は一般的に「get + ＜フィールド名＞」や「set + ＜フィールド名＞」の形をとることが多く，命名支援が必要となるほど複雑な命名になることは少ないと考えられる．しかし，このような”get”や”set”を含むメソッドは，提案手法のデータセットに含まれる 125,199 件のメソッドのうち，約 20 %に相当する 24,457 件が該当する．また，比較手法のデータセットに含まれる 117,652 件のメソッドのうち，約 29 %に相当する 33,835 件が該当する．よって，これらを評価の対象に含めるか含めないかで評価を行うことで精度に大きな違いが生じる可能性があるため，項目 1 でこれらを含めて評価を行うと同時に，この項目ではこれらを含めずに評価を行う．

3. 動詞ごとに予測精度を評価する.項目 1 と同様にテストデータに含まれる全メソッド宣言について評価を行うが，これらをメソッド名に含まれる動詞単語ごとにグループ分けし，動詞ごとに予測成否の割合を算出する．これにより，提案手法や比較手法がどのような動詞を含むメソッド名に対して有効であるのかを評価する．

表 2　動詞部分の予測結果（5 分割交差検証）

	提案手法		比較手法	
	メソッド数	精度	メソッド数	精度
subset 1	16,426	49.91%	15,968	53.84%
subset 2	17,639	51.17%	16,140	53.27%
subset 3	18,076	49.93%	17,836	54.73%
subset 4	16,844	49.74%	15,561	57.10%
subset 5	17,759	51.66%	17,127	52.38%
平均	50.48%		54.26%	

表 3　名詞部分の予測結果（5 分割交差検証）

	提案手法		比較手法	
	メソッド数	精度	メソッド数	精度
subset 1	15,903	46.41%	16,853	65.60%
subset 2	16,669	45.94%	17,097	63.40%
subset 3	17,669	44.67%	18,828	66.07%
subset 4	15,647	46.48%	16,096	66.66%
subset 5	17,382	47.38%	18,008	66.04%
平均	46.17%		65.55%	

4.1　実験結果および考察

本小節では先述した各評価項目に関してそれぞれ実験結果を述べた後，考察する．

4.1.1　テストデータ中の全てのメソッドに関する評価

本節では評価項目 1 に関して述べる．表 2 は 5 分割交差検証の各サブセットにおける，提案手法と比較手法の動詞部分の予測精度を示したものである．表 3 は同様に名詞部分の予測精度を示したものである．評価用データに含まれるメソッド名の中には，品詞判定の結果動詞を含まないと判定されたもの，または名詞を含まないと判定されたものが存在するため，それらは評価の対象外とした．表 2 と表 3 では，提案手法と比較手法それぞれに関して，精度を測定した評価対象のメソッドの数を精度と共に表記した．これらの表から，テストデータ中全てのメソッドを対象とした場合，提案手法は動詞部分・名詞部分の予測精度両方に関して提案手法は比較手法に劣るという結果となった．

4.1.2　単純なメソッド名のメソッドを除外して評価

本節では評価項目 2 について述べる．本項目では，5 分割交差検証の各サブセットのテストデータから，メソッド名に”get”，”set”を含むメソッドをすべて除いた状態で評価項目 1 と同様に予測精度を算出した．表 4 は動詞部分の予測精度を，表 5 は名詞部分の予測精度を示したものである．表 4 と表 2 を比較した結果，getter や setter といった単純な名前のメソッドを評価から除いた場合，除かなかった場合と比較して提案手法はメソッド名の動詞部分に関して比較手法よりさらに良い精度で予測ができることが分かる．同様に，表 5 と表 3 を比較した結果，getter や setter といった単純な名前のメソッドを評価か

表 4　動詞部分の予測結果
（5 分割交差検証，”get”,”set”を含むメソッドを除外）

	提案手法		比較手法	
	メソッド数	精度	メソッド数	精度
subset 1	11,909	42.51%	9,378	28.65%
subset 2	12,509	45.10%	9,374	30.67%
subset 3	12,975	42.74%	10,404	31.28%
subset 4	11,850	42.04%	8,749	33.17%
subset 5	13,004	44.27%	10,892	33.91%
平均	43.36%		31.53%	

表 5　名詞部分の予測結果
（5 分割交差検証，”get”,”set”を含むメソッドを除外）

	提案手法		比較手法	
	メソッド数	精度	メソッド数	精度
subset 1	11,725	46.30%	10,533	56.16%
subset 2	11,895	46.77%	10,607	52.56%
subset 3	12,907	45.47%	11,667	56.58%
subset 4	10,985	47.25%	9,544	56.24%
subset 5	12,988	46.97%	12,046	57.84%
平均	46.55%		55.88%	

ら除いた場合でも，名詞部分の予測精度に関しては提案手法は比較手法に劣るが，精度の差は除かなかった場合と比較して小さくなることが分かる．以上の結果から，提案手法は getter や setter といった単純な命名のメソッド以外に関しては比較手法より優れた精度で名前の予測ができる可能性があると考えられる．対して，比較手法は getter や setter といった単純な命名のメソッドに関しては提案手法より優れた精度で名前の予測ができる可能性があると考えられる．

4.1.3　動詞ごとに予測精度を評価

本節では評価項目 3 について述べる．本項目ではテストデータに含まれているテスト用メソッド宣言の集合を，メソッド名に含まれている動詞ごとにグループ分けし，各動詞ごとに提案手法が比較手法と比べてどの程度予測精度が変化しているかを算出した．図 3 はその動詞が含まれているメソッドが 5 個未満の動詞群に関して，横軸は比較手法に対する提案手法のその動詞に関する予測制度の上昇値，縦軸にその上昇値に該当する動詞の数をプロットしたものである．同様に，図 4 から図 7 はそれぞれその動詞が含まれているメソッドが 5 個以上 10 個未満，10 個以上 20 個未満，20 個以上 100 個未満，100 個以上の動詞に関してプロットしたものである．該当する動詞の総数は図 3 から図 7 までそれぞれ 1,067 個，203 個，180 個，220 個，101 個である．これらの図から，5 個未満のメソッドにしか含まれないような動詞に関しては，予測精度が変わらない動詞が多数を占め，また予測精度が上昇した動詞の数と減少した動詞の数がほぼ変わらないことが分かる．5 個以上 10 個未満のメソッドに含まれるようなある程度頻繁に使用されるような動詞に関しては，予測精度が上昇した動詞の数と減少した動詞の数がほぼ同等であることが分かる．10 個以上のメソッドに含まれるような頻繁に使用されるような動詞に関しては，予測精度が上昇した動詞の数が減少した動詞の数と比較して多いことが分かる．以上の結果から，10 個未満のメソッドにしか含まれないようなマイナーな動詞に関しては提案手法と比較手法では予測精度がほぼ変わらないと言える．100 個以上のメソッドに含まれるような比較的メジャーな動詞に関しては，提案手法のほうが比較手法と比較して予測精度が良いといえる．

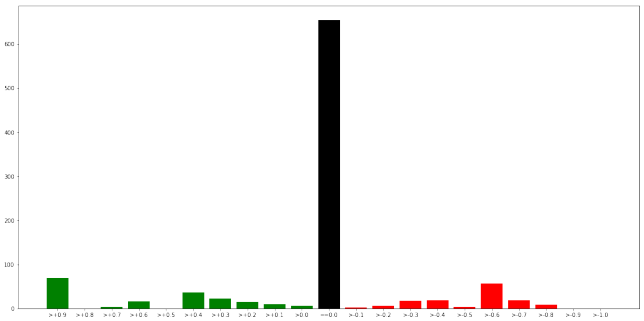

図 3　提案手法と比較手法の予測精度の差
（その動詞が含まれているメソッドが 5 個未満の動詞）

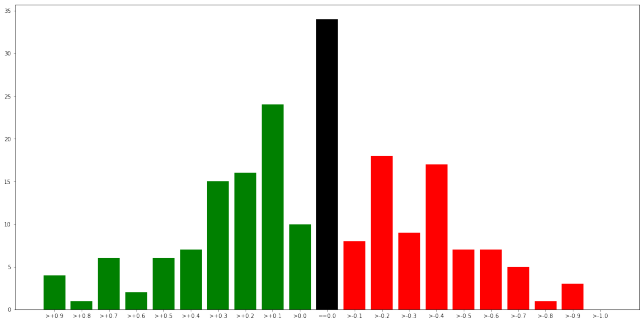

図 4　提案手法と比較手法の予測精度の差
（その動詞が含まれているメソッドが 5 個以上 10 個未満の動詞）

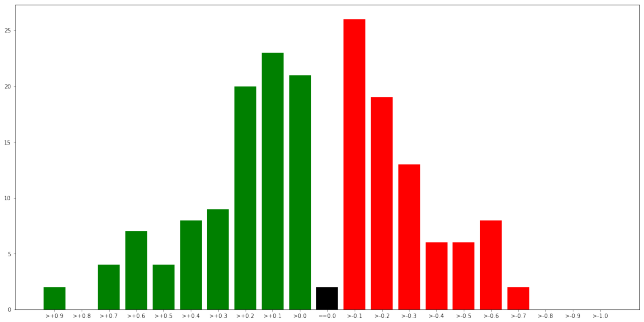

図 5　提案手法と比較手法の予測精度の差
（その動詞が含まれているメソッドが 10 個以上 20 個未満の動詞）

5　考　　察

データセット全体を対象とした実験では，提案手法は比較手法より動詞・名詞共に予測精度が低かった．この実験は getter や setter の役割をもつと考えられるメソッドを含むものであるが，このようなメソッドは，そのクラスの何かしらのフィールドと 1 対 1 で対応していることが多く，そのメソッドの内部で参照しているフィールドの名前がメソッド名を予測する上で大

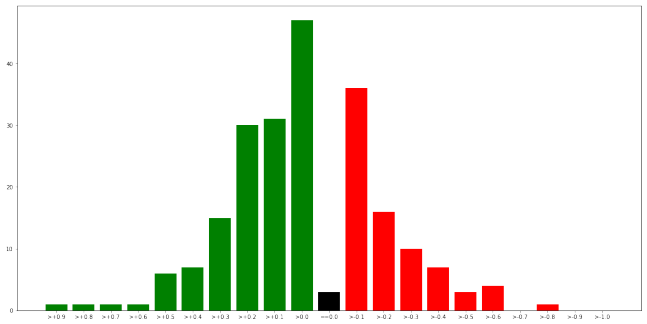

図 6　提案手法と比較手法の予測精度の差
（その動詞が含まれているメソッドが 20 個以上 100 個未満の動詞）

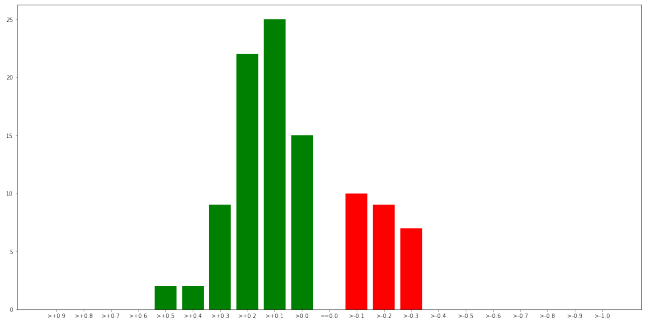

図 7　提案手法と比較手法の予測精度の差
（その動詞が含まれているメソッドが 100 個以上の動詞）

きな手がかりとなり得る．比較手法は入力のトークン列の中に内部で参照しているフィールドの名前が含まれるため，これが正しいメソッド名の予測に貢献している可能性があるが，提案手法は内部で呼び出しているメソッド名しか利用していないため，参照しているフィールドの名前を手がかりとして利用できなかったことが予測精度に差が出た理由として考えられる．一方で，getter，setter を除いた実験では提案手法は比較手法より動詞の予測精度が高いという結果であった．これより，予測対象のメソッドの振る舞いや動作に応じて，提案手法による予測と比較手法による予測を組み合わせることで，全体としての予測精度を向上できる可能性があると言える．また，提案手法は比較的メジャーに使用される動詞に関しては比較手法より予測精度が高かったため，提案手法によって予測したメソッド名の候補に含まれる動詞が，メソッド名としてどの程度メジャーに用いられるかに応じて，提案手法による予測と比較手法による予測を組み合わせることで，全体としての予測精度を向上できる可能性があると言える．

本実験では提案手法と比較手法で同じデータセットを使用したが，抽出したメソッドの数は異なるものであった．特に比較手法で抽出したメソッド集合は，その約 3 割が getter や setter と思われるメソッドであったため，そもそもデータセットが実験に適していなかった可能性がある．

6　結　　論

本研究では，メソッド名の命名が難しいという問題に対し，メソッドの定義が既に存在するようなメソッドを改名するような場面を対象として，そのメソッドの名前を予測，推薦する手法を提案した．予測に際して，メソッドのメソッド名とその定義で呼び出されているメソッド集合の関係を利用した．また，収集したメソッドの呼び出し関係に基いて，各メソッド名に関する分散表現を生成し，予測に利用した．

本稿では，提案手法と既存手法に関して，メソッドの名前を予測する実験を行い，予測精度を比較・評価した．その結果，提案手法はメソッド名に対する命名支援が必要となるような複雑なメソッド名の動詞部分に関して，既存手法より優れた精度で予測が可能であった．また，多くのメソッドで用いられるようなメジャーな動詞に関して，既存手法より優れた精度で予測が可能であった．

今後の方針として，今回は単純化のためメソッド定義の内部で呼び出しているメソッドのみに注目したが，変数や引数の型といった呼び出しメソッド以外の要素も利用して分散表現の生成に利用することが考えられる．また，名前予測するメソッドの特徴に応じて既存手法による予測と提案手法による予測を組み合わせることで，全体として予測精度を向上させるような手法を検討することが挙げられる．また，より実用的な推薦手法として，メソッド名を構成単語に分割し別々に推薦を行うような推薦手法を検討すること，実際の開発の現場で利用できるような推薦システムを検討することが挙げられる．

謝　　辞

本研究は，平成 30 年度共同研究 (SKY 株式会社)（CPE30034）「データエンジニアリングの知見の応用による SKYSEA Client View のログ及び資産情報の処理の高速化・軽量化・高度化」の助成を受けたものである．

文　　献

[1] Bennet P Lientz, E. Burton Swanson, and Gail E Tompkins. Characteristics of application software maintenance. *Communications of the ACM*, Vol. 21, No. 6, pp. 466–471, 1978.

[2] Gail C Murphy, Mik Kersten, Martin P Robillard, and Davor Cubranic. The emergent structure of development tasks. In *ECOOP*, Vol. 5, pp. 33–48. Springer, 2005.

[3] Andrea De Lucia, Massimiliano Di Penta, Rocco Oliveto, Annibale Panichella, and Sebastiano Panichella. Using ir methods for labeling source code artifacts: Is it worthwhile? In *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on*, pp. 193–202. IEEE, 2012.

[4] 柏原由紀. メソッド名とその周辺の識別子の相関ルールに基づくメソッド名変更支援手法. 大阪大学基礎工学部情報科学科特別研究報告, 2013.

[5] Yuki Kashiwabara, Yuya Onizuka, Takashi Ishio, Yasuhiro Hayase, Tetsuo Yamamoto, and Katsuro Inoue. Recommending verbs for rename method using association rule mining. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on*, pp. 323–327. IEEE, 2014.

[6] Miltiadis Allamanis, Earl T Barr, Christian Bird, and Charles Sutton. Suggesting accurate method and class names. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pp. 38–49. ACM, 2015.

[7] 米内裕史, 早瀬康裕, 北川博之. メソッド呼び出し関係に基づくメソッド名の予測. ソフトウェアエンジニアリングシンポジウム 2018 論文集, Vol. 2018, pp. 34–43, 2018.

[8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, Vol. 3, No. Feb, pp. 1137–1155, 2003.

[9] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *hlt-Naacl*, Vol. 13, pp. 746–751, 2013.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[12] Zellig S Harris. Distributional structure. *Word*, Vol. 10, No. 2-3, pp. 146–162, 1954.

[13] Einar W Host and Bjarte M Ostvold. The programmer's lexicon, volume i: The verbs. In *Source Code Analysis and Manipulation, 2007. SCAM 2007. Seventh IEEE International Working Conference on*, pp. 193–202. IEEE, 2007.

[14] Einar W Høst and Bjarte M Østvold. Debugging method names. In *European Conference on Object-Oriented Programming*, pp. 294–317. Springer, 2009.

[15] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, Vol. 22, pp. 207–216. ACM, 1993.

[16] Miltiadis Allamanis, Hao Peng, and Charles Sutton. A convolutional attention network for extreme summarization of source code. In *International Conference on Machine Learning (ICML)*, 2016.

[17] Suggesting accurate method and class names. `http://groups.inf.ed.ac.uk/cup/naturalize/`.

[18] Github. `https://github.com/`.