処理目的と対応するプログラムパターンの自動収集

間嶋　義喜[1,a)]　廣川　佐千男[2,b)]　竹内　和広[3,c)]

概要：本稿では，プログラムの木構造に対するグラフマイニング手法を提案し，マイニングで得られた構造的特徴とプログラムの部分機能を説明する自然言語との対応を検討する．具体的には提案手法により，自然言語で記述された課題に対して作成されたプログラムが，9 種類中のどの課題のために作成されたものかを，構造的特徴のみから判別できることを確認する．次に判別に貢献したプログラムの構造的特徴がどのような自然言語表現と対応付くかを検討した．その結果，特定のメソッド名の命名に使用される自然言語表現に対応付くプログラムの特徴的構造があることを確認できた．

キーワード：プログラム解析，プログラムパターン，Support Vector Machine

Examination of program structure extraction for processing contents estimation

Abstract: In this paper, we propose a graph mining method for the tree structure of a program, in order to find structural features that can be corresponded to natural language words expressing partial functions of programs. As the first step to the goal, we confirmed that our proposed method can distinguish the programs that are created for a particular task assignment from the others, based only on the structural features. The data for the confirmation is a set of programs that are made for nine task assignments described in natural language. Based on the confirmation, we examined what kind of natural language word corresponds to structural features of the program that contributed to the discrimination of the particular task assignment. Through the investigation, we confirmed that there are structural features in programs corresponding to natural language word used for naming specific computational methods.

Keywords: program analysis, program pattern, support vector machine

1. はじめに

ソフトウェア開発やプログラミング教育の分野において，プログラムの開発・分析や採点には多大な労力と時間がかかる．プログラムの分析には，クラス名やメソッド名等の識別子に使われる自然言語が参考になる．しかし，自然言語の使用には表記の揺れの問題が存在する．例えば，あるプログラマが iteration というメソッド名を命名した部分プログラムに対して，iter と略語で命名するプログラマもいれば，repeat というメソッド名を命名するプログラマもいる．ソフトウェアプロジェクトにおいて識別子の命名を共有することは，このような自然言語使用の揺れの問題を解消する知恵と考えることもできる．

我々は，プログラムの部分構造に対するコメント・説明を生成，あるいは自然言語からのプログラム生成や検索を行う基盤となる，プログラム構造と自然言語の対応付けの整理を目指している．

本稿では，プログラム部分の目的や機能を自然言語で記述したものを，便宜上，処理目的と呼ぶ．処理目的は本来，複数の語から構成され，文章や文によって記述されると考えられるが，文，段落，文章と，それらの単位を構成する語数が増えるほど多様性が大きくなる．

本稿ではまず，自然言語とプログラム構造との対応付け

[1] 大阪電気通信大学大学院
Osaka Electro-Communication University
[2] 九州大学情報基盤研究開発センター
Research Institute for Information Technology Kyushu University
[3] 大阪電気通信大学
Osaka Electro-Communication University
[a)] mi17a004@oecu.jp
[b)] hirokawa@cc.kyushu-u.ac.jp
[c)] takeuchi@osakac.ac.jp

の基礎となる，プログラムの木構造に対するグラフマイニング手法を提案し，提示された課題を解くプログラム集合（Task プログラム）を評価データに，プログラムの構造的特徴（プログラムパターン）から，作成されたプログラムの課題を同定できるかを検討する．ここで，分析対象となるプログラムのそれぞれの処理目的は，自然言語で記述された課題文としている．このことから，我々の提案するマイニング手法が抽出可能なプログラムの静的な構造的特徴から，処理目的（ここでは課題文）をどの程度判別可能かを見積もりたい．

次に，上記で有効性を検討したグラフマイニング手法を，より大規模なプログラム集合に対して適用し，特定のメソッド名の判別に強く関わるプログラムパターンを抽出することにより，プログラムパターンとメソッド名に使用される自然言語の語との対応関係を確認する．

2. 関連研究

Gvero ら [1] は，自然言語記述からプログラムで実行可能な関数呼び出し文を自動生成する，anyCode と呼ばれるシステムを開発している．このシステムは，自然言語とプログラムとの対応付けを行うことによって実現されている．anyCode は，開発者が処理を行いたい機能を持つ関数呼び出し文やその使い方を知らなくても，自然言語による問い合わせによって，実行可能な関数呼び出し文を生成することができる．

分析対象となるプログラムの扱い方は大きく 2 種類あり，記号列のまま扱う場合と数値表現に変換して扱う場合がある．プログラムを記号列として捉えることで，自動修正や修正箇所を検知したり [2],[3],[4]，シーケンシャルパターンの活用による潜在的なバグ検出 [5] なども試みられている．一方で数値表現として捉えることで，プログラム内で類似または一致する記述（クローン）の検出やプログラムの分析に貢献する試みがなされている．村上らは，状態ベクトルと呼ばれる AST に基づくプログラムの数値化を行うことで，プログラムのバージョンアップに伴うプログラムの変化を予測する手法を提案し，プログラムの修正すべき箇所の特定に貢献している [6]．Jiang らは，AST に基づく非終端記号の出現回数に着目したプログラムの数値化を提案し，Locality Sensitive Hashing によりクローンの検出を行なっている [7]．Peng らは，プログラムを AST で使用される非終端記号で表現し，プログラム群を表現するベクトル空間を獲得した後，機械学習を活用した AST のノード間の関係調査やプログラムの分類を試みている [8]. 山本らは Doc2Vec と呼ばれる，文書を単位とした分散表現を得る技術を活用し，プログラムのベクトル表現の獲得を試みている [9]．これらベクトル空間は分散表現と呼ばれ，その獲得に Neural network を利用している．分散表現は，自然言語処理の分野で提案された，単語や文をベクトル空間上に表現する手法であり，単語間の関係が密になるようにベクトル空間を構築する．このようなプログラム構造の数値化や分散表現などの応用は，プログラム構造における特徴的なパターンの抽出において，我々の研究における方向性と共通している．

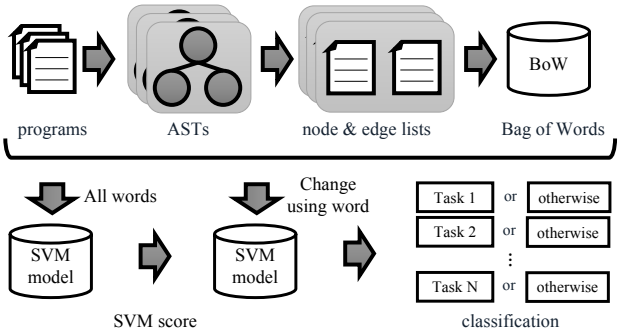


図 1: SVM を利用したプログラムの識別手法

3. SVM によるプログラムの識別

プログラムの処理目的と自然言語の対応付けに向けて，本稿では，プログラムの処理目的を提示された課題内容とし，その課題内容に投稿されたプログラム集合から課題ごとにプログラムパターンを抽出することで，課題内容から推定される機能の検討を行う．部分構造の抽出には，SVM^{perf}[10] で実装されている Support Vector Machine（SVM）を活用し，廣川らが提案した手法を採用する [11],[12]．SVM^{perf} は，サポートベクトルに基づいた使用する素性の重要度を示す評価指標（SVM スコア）が実装されている．この指標を活用することで課題ごとに重要とされる素性を組み合わせ，特徴的な部分構造を抽出する．廣川らは，素性ごとに計算される SVM スコアの正と負の上位をペアとして順に使用することで，識別に寄与する素性の選択を行っている．本稿で行う実験概要を図 1 に示す．

課題ごとに最適な素性を抽出するために，投稿されたプログラムがどの課題内容を解くためのプログラムかという SVM モデルを構築し，F 値に基づく最適な素性数を計測する．すなわち，2 値分類における機械学習モデルを評価する precision，recall，f-measure，accuracy を 10 分割交差検証によって計測する．

3.1 Support Vector Machine

Support Vector Machine(SVM)[13] は，主に 2 クラス分類問題を解く手法として考案され，未知データに対して高い予測精度を持つ分類器を構築できることが報告されている．n 個の事例からなる訓練集合 $\{(\boldsymbol{x}_i, y_i)\}_{i\in[n]}$ が，d 次元実数ベクトル $\boldsymbol{x}_i \in \mathbb{R}^d$ と，1 か -1 の値を取るラベル $y_i \in \{-1, 1\}$ から構成されているとする．SVM は $\boldsymbol{x}$ の空間において，二つのクラスが分離境界からどのくらい離れ

表 1: プログラム行数に着目した Task プログラムの規模

Task	ファイル数	平均値	中央値	最頻値	Total LOC
1	300	23.03	20	15	6,909
2	300	16.08	13	11	4,823
3	300	19.22	18	15	5,766
4	300	25.31	24	26	7,592
5	300	23.42	20	20	7,026
6	300	23.35	21	18	7,005
7	300	30.27	28	28	9,080
8	300	25.95	23	21	7,784
9	300	24.92	22	21	7,464

ているか，すなわちマージンを最大化することによって，分離境界を決定する．マージン最大化は，分離境界と分離境界から最も近くにある $\boldsymbol{x}$ ベクトルとの距離を最大化することによって得られる．このマージン最大化に寄与する $\boldsymbol{x}$ ベクトルをサポートベクトルと呼ぶ．

分類可能な訓練集合に対して制約条件を緩和したソフトマージンの最適化問題は式 (1) で定義され，これを解くことで分類器を得ることができる．ここで，ξ は誤分類の調整に使用されるパラメータであり，C は正則化係数である．

$$\min_{\boldsymbol{\omega},b,\xi} \quad \frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{i\in[n]}\xi_i \tag{1}$$

$$\text{s.t.} \quad y_i(\boldsymbol{\omega}^T\boldsymbol{x}_i + b) \geq 1 \tag{2}$$

$$\xi_i \geq 1,\ i \in [n]$$

3.2 対象とした Task プログラム

本稿では Aizu Online Judge（AOJ）[14],[15] と呼ばれる，オンライン上で出題される各課題に対して投稿された Task プログラムを使用する．AOJ は，投稿されたプログラムをテストケースによる正誤判定やプログラムの実行効率などから自動的に評価するシステムである．Task プログラムの規模を確認するために，プログラムの行数に着目した Task プログラムの規模を表 1 に示す．Total LOC は，各 Task に提出されたプログラム行数の合計値を示し，課題内容の一例を表 2 に示す．

3.3 プログラムの数値変換

プログラムは，規定されたプログラミング言語の文法に従い，記号列で記述される．そのため，プログラムパターンを抽出するために，文法的な構造を捉えた表現に変換することが必要である．本稿では，抽象構文木（Abstract Syntax Tree, AST）で構成される構造的特徴に着目して，プログラムの特徴表現を得る．AST とは，プログラミング言語に基づく文法から構築され，プログラムの持つ構文関係を木構造で表現する．ノードは文法に規定される非終端記号であり，そのノード同士の関係性をエッジによって表現する．本稿では，プログラムは構文規則の組み合わせにより構築されるという概念に基づき，AST に基づくプログラムの部分構造を素性とする．具体的には，構文解析により Task プログラムから長さ 5 以下の連結されたノードのパスを収集し，それらを素性としたベクトルでプログラムを表現する．ベクトルの要素は，素性が出現するかどうかという真理値を持つ．

3.4 識別結果

Task ごとに投稿されたプログラムが，どの Task を解くためのプログラムであるかという識別結果を，表 3 に示す．N 列は，SVM スコアに基づいて使用する素性数を変化させた場合における，F 値に基づく最適な属性数を示す．Accuracy 列の範囲は，0.9642–0.9985 であり，F-measure 列の範囲は，0.8440–0.9933 と高い識別結果であることが確認される．この結果から，本実験で採用した手法とプログラムの数値表現は，規模の小さなプログラム集合の識別において有効であることが確認される．

4. 抽出された構造パターンの確認

本節では，識別に寄与する素性を組み合わせた素性のグラフ化（特徴語グラフ）を行い，内包している機能を持つメソッド名を考察する．

グラフ化するために使用した素性は，エッジを持つ 2 以上 5 以下のパスであり，エッジごとのスコア（エッジスコア）が最大値の 50%以上のスコアを持つエッジをグラフ（特徴語グラフ）として表示する．またエッジスコアは，SVM スコアの合計値とする．例えば，ForStatement–ExpressionStatement–MethodInvocation–MethodInvocation–MethodInvocation という素性があり，その SVM スコアが 0.0768 であるとする．この時，同じノード名が一つの素性の中に複数出現する場合，ノードの区別は行わず，同じパスの中におけるエッジの出現回数の乗数を取る．よってエッジスコアは，ForStatement–ExpressionStatement と ExpressionStatement–MethodInvocation は 0.768，MethodInvocation–MethodInvocation は 0.768^2 となる．これらを各素性ごとに算出し，同じ種類のエッジで合計値を取る．

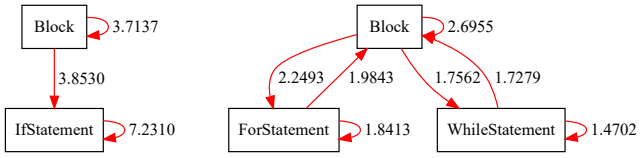
Task7,9 の特徴語グラフを図 2 に示す．Task7 では，IfStatement–IfStatement 構造という多岐条件分岐による構造パターンが確認される．この Task は，2 つの整数と演算子が入力され，該当する演算を探すときに多岐条件分岐が使用されると考えられるため，search などのメソッド名が該当すると考えられる．Task9 では，ForStatement–ForStatement 構造や WhileStatement–WhileStatement 構造という二重に組み合わされた繰り返し構造パターンが確認される．この Task は，入力される 2 つの整数から記号で矩形を描画する際に二重の繰り返し構造が使用されると

表 2: 課題内容の一例（Task7 と Task9）

Task	課題内容
7	2 つの整数 a, b と 1 つの演算子 op を読み込んで，$a\ op\ b$ を計算するプログラムを作成せよ．ただし，演算子 op は，”+”(和)，”−”(差)，”∗”(積)，”/”(商) のみとし，割り算で割り切れない場合は，小数点以下を切り捨てたものを計算結果とする．入力は複数のデータセットから構成される．各データセットの形式は，$a\ op\ b$ となり，op が”?”のときに入力の終わりを示す．出力は，各データセットについての計算結果を 1 行に出力せよ．
9	たて H(cm)，よこ W(cm) の長方形を”#”で描くプログラムを作成せよ．1cm × 1cm の長方形を”#”で表せ．入力は複数のデータセットから構成されている．各データセットの形式は，$H\ W$ となっている．H, W がともに 0 のとき，入力の終わりとする．出力は各データセットについて，$H \times W$ 個の”#”で描かれた長方形を出力せよ．また各データセットの後に，1 つの空行を入れること．

表 3: SVM によりプログラム分類を行なった識別結果

Task	N	Precision	Recall	F-measure	Accuracy
1	2000	0.9279	0.9342	0.9306	0.9845
2	4000	0.9941	0.9684	0.9810	0.9957
3	4000	0.9625	0.9482	0.9550	0.9900
4	4000	0.8213	0.8684	0.8440	0.9642
5	3000	0.9580	0.9769	0.9672	0.9926
6	5000	0.9307	0.9257	0.9278	0.9841
7	3000	0.9871	0.9759	0.9813	0.9960
8	4000	0.9827	0.9573	0.9697	0.9933
9	4000	0.9968	0.9898	0.9933	0.9985



(a) (b)

図 2: SVM スコアに基づく各 Task の識別に寄与する特徴語グラフ (a)Task7 (b)Task9

考えられるため，traversal や trace などのメソッド名が該当すると考えられる．

このような関係の裏付けとして，実践的なプログラム集合である Allamanis らが収集したコーパス [16] において，出現頻度が高い equals, get, toString などのメソッド語とプログラムの構造的特徴との強い対応が確認できている．現在は，より大規模に調査を進めているところである．

5. おわりに

本稿では，プログラム構造と自然言語の対応付けを整理する上で必要となるプログラムパターンの収集に向けて，SVM を活用したグラフマイニング手法を提案した．その結果，AST というプログラムの静的な表現に限定した構造的特徴により，高い精度で AOJ の該当課題を識別し，プログラムパターンを抽出することができた．

今後，静的な表現で得られたプログラムパターンの整理・分析を前提に，プログラムの動的側面も考慮したプログラム分析を展開していきたい．

参考文献

[1] Gvero, T. and Kuncak, V.: Synthesizing Java Expressions from Free-form Queries, *In Proc. OOPSLA*, pp. 416–432 (2015).
[2] Le, G. C., Nguyen, T., Forrest, S. and Weimer, W.: Genprog: A generic method for automatic software repair, *IEEE Transactions on Software Engineering*, pp. 54–72 (2012).
[3] Nguyen, H. D. T., Qi, D., Roychoudhury, A. and Chandra, S.: Semfix: Program repair via semantic analysis, *In Proc. ICSE*, pp. 772–781 (2013).
[4] Mechtaev, S., Yi, J. and Roychoudhury, A.: Directfix: Looking for simple program repairs, *In Proc. ICSE*, pp. 448–458 (2015).
[5] Kagdi, H., Collard, M. L. and Maletic, J. I.: Comparing approaches to mining source code for call-usage patterns, *In Proc. MSR*, pp. 123–130 (2007).
[6] Murakami, H., Hotta, K., Higo, Y. and Kusumoto, S.: Predicting Next Changes at the Fine-Grained Level, *In Proc. APSEC*, pp. 119–126 (2014).
[7] Jiang, L., Misherghi, G., Su, Z. and Glondu, S.: Deckard: Scalable and accurate tree-based detection of code clones, *In Proc. ICSE*, pp. 96–105 (2007).
[8] Peng, H., Mou, L., Li, G., Liu, Y., Zhang, L. and Jin, Z.: Building Program Vector Representations for Deep Learning, *In Proc. KSEM*, pp. 547–553 (2015).
[9] Yamamoto, T.: A Study on Source Code Search and Classification using Distributed Representations, *IEICE technical report*, pp. 67–72 (2017).
[10] Joachims, T.: Training linear SVMs in linear time, *In Proc. KDD*, pp. 217–226 (2006).
[11] Sakai, T. and Hirokawa, S.: Feature Words that Classify Problem Sentence in Scientific Article, *In Proc. iiWAS*, pp. 360–367 (2012).
[12] Adachi, Y., Onimura, N., Yamashita, T. and Hirokawa, S.: Standard measure and SVM measure for feature selection and their performance effect for text classification, *In Proc. iiWAS*, pp. 262–266 (2016).
[13] Cortes, C. and Vapnik, V.: Support vector networks, *Machine learning*, pp. 273–297 (1995).
[14] Watanobe, Y.: Aizu Online Judge, http://judge.u-aizu.ac.jp/onlinejudge/.
[15] 渡部有隆：オンラインジャッジの開発と運用 -Aizu Online Judge-，情報処理，pp. 998–1005 (2015).
[16] Allamanis, M. and Sutton, C.: Mining Source Code Repositories at Massive Scale Using Language Modeling, *In Proc. MSR*, pp. 207–216 (2013).