

学習者の作成進度に応じた個別対応のためのプログラム表現学習

竹末 裕^{1,a)} 竹内 和広^{2,b)}

概要: 学習者のプログラム課題実施中に、計算機が学習者の躓きや迷いに応じたアドバイスを提供するシステムを構築したい。そのような場合、作成途中にあるプログラムが、課題の達成点に関して、どの程度の到達度にあるかを判定することが課題となる。本稿では、作成途中にあるプログラムであっても抽象構文木で扱う手法を提案するとともに、AOJ(Aizu Online Judge) の完成プログラム集合からあらかじめ抽出・整理したプログラムの学習ポイントを対応付け、学習者が作成中のプログラムに対して完成からの差分に基づいて評価が行える手法を紹介する。具体的には、プログラムの編集過程を考慮して抽象構文木をベクトル空間に埋め込む表現学習を用い、プログラムと作成目的との対応付けについて検討を行った。

Representation learning for analyzing programming progress states of learners

YUTAKA TAKESUE^{1,a)} KAZUHIRO TAKEUCHI^{2,b)}

1. はじめに

プログラミング学習はプログラム言語の学習とプログラム言語によってデータ構造やアルゴリズムを学ぶことが基本となる。しかし、目的となる問題解決を行うためには、複数の基本的なデータ構造やアルゴリズムを改変したり、組み合わせたりすることが必要である。そのような問題解決のプログラムに関する応用力の練成には、多くの大学で、プログラムの座学だけではなく演習を設定しているように、具体的な問題に対して試行錯誤を行いつつ実践的に学習することが効果的であることを示している。

本稿では、問題解決のために、数々のプログラムの部分目的を達成するために使われるプログラムの構造的類型をプログラムパターンと呼ぶ。学習者のプログラムからプログラムパターンを発見することは簡単ではない。なぜなら、プログラムの学習者が問題解決に詰まり迷いが生じる

のは、プログラム作成過程であり、目的の動作するプログラムとはプログラムの文法に従っているとは限らない。また、動作さえしないプログラムである場合もありうる。このような学習者の編集途中のプログラムを処理するため、本稿では AST (Abstract Syntax Tree) を基本とした木構造で表現する。

さらに、本稿では、プログラムの類型を計算する上で表現学習を用いる。表現学習は、画像、音、自然言語、時系列データなどの高次元なデータを低次元な特徴空間で表現するモデルを作成する機械学習の手法であり、特定の目的に有用な特徴表現モデルを獲得する手法とも捉えることができる。

2. 分散表現

2.1 文章の分散表現

本稿では、文章のベクトル表現を得る Doc2Vec[3] を用いる。Doc2Vec は、語を扱う Word2vec[4][5] を参考に拡張した、NN(ニューラルネットワーク) を題材に提案された手法の一つであり、文章から固定長の特徴ベクトルを生成するモデルである。具体的には、Doc2Vec は、Mikolov ら

¹ 大阪電気通信大学大学院
Osaka Electro-Communication University

² 大阪電気通信大学
Osaka Electro-Communication University

^{a)} mi19a004@oecu.jp

^{b)} takeuchi@osakac.ac.jp

によって提案された単語の特徴ベクトルを生成するモデルである Word2vec の周辺語から特定の語を予想するモデルを、特定の文章における周辺語から特定語の予想をするモデルに拡張している。すなわち、Doc2Vec を NN で実装する場合の構造は、文書中のある単語に注目し、注目単語の前後関係を NN を用いて学習させるという形をしており、ほぼ Word2vec と同様の構造を持っている。

Word2vec と Doc2Vec で本質的な違いは、後者が単語 ID に加え、扱う文章集合中の特定文章を示すパラグラフ ID(文章 ID) を付加した情報を入力とする。また、Doc2Vec は Word2vec と同様に、分散型メモリ (DM) モデルと分散型の語彙 (DBOW) モデルの二種類の学習モデルが存在している。ここで用いられる NN は、単純な順伝播型ネットワーク (feedforward neural network 以下、FNN) である。FNN は、情報が入力層から出力層に一方方向にのみ伝搬していくネットワークであり、多層パーセプトロン MLP(Multilayer perceptron) とも呼ばれる。MLP はパラメータを調整することにより、様々な関数に近似することが出来る [2][1]。

DM モデルは、Word2vec の CBOW モデルの入力層に文書ベクトルを付け加えて NN に学習させる手法である。学習する NN の形を図 1 に示す。

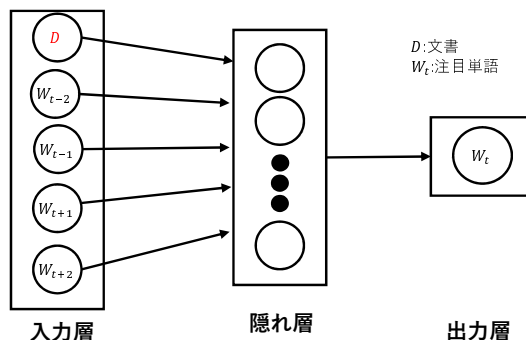


図 1 DM モデルのイメージ図

DBOW モデルは、Skip-gram モデルの入力層の単語 ID を文章 ID に変換させたモデルである。つまり、文章 ID から、その文章に含まれている単語を教師値として、予測させるモデルである。学習する NN の形を図 2 に示す。

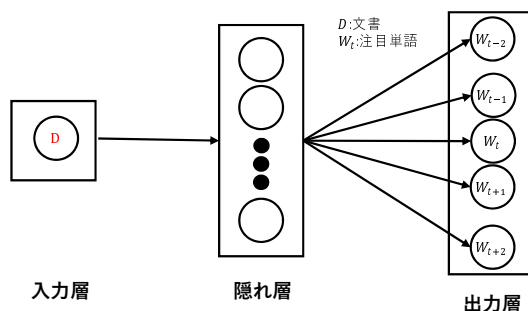


図 2 DBOW モデルのイメージ図

2.2 グラフの表現学習

graph2vec[6] は文章の分散表現を学習する Doc2vec を参考に、グラフ埋め込みを学習するための拡張を行っている。Doc2vec は、単語や単語シーケンスが文章を構成する方法を利用して、埋め込みを学習している。同様に、graph2vec では、グラフ全体を文章として捉え、グラフ内のすべてのノードの接続先を考慮した根付き部分グラフを単語として捉え、グラフ全体の表現を学習することを提案したものである。graph2vec は、グラフ全体の表現を学習する最初の NN の埋め込みであり、グラフカーネルやその他の下位構造の埋め込みに比べて次の特徴がある。

(1) 教師なし表現学習

教師なしの学習でグラフ埋め込みを行う。つまり、グラフのクラスラベルは、グラフ埋め込みの学習に必要な。これにより、ラベル付けされたデータを取得するのが難しい多数のアプリケーションで、graph2vec 埋め込みを簡単に使用できる。

(2) タスクに依存しない埋め込み

graph2vec は、表現学習プロセスのためにタスク固有の情報(クラスラベルなど)を利用しないため、graph2vec が提供する埋め込みは一般的である。これにより、グラフ全体を含むすべての分析タスクでこれらの埋め込みを使用できる。

(3) データ駆動型の埋め込み

グラフカーネルとは異なり、グラフデータの大規模なコーパスからグラフ埋め込みを学習する。これにより、手作りの機能ベースの埋め込みアプローチの欠点を回避できる。

(4) 構造的同等性をキャプチャ

グラフ内の線形部分構造(固定長のランダムウォークなど)をサンプリングしてグラフ埋め込みを学習する sub2vec などのアプローチとは異なり、フレームワークは非線形部分構造、つまり埋め込みを学習するためのルート化された部分グラフをサンプリングして考慮している。

このような非線形部分構造が構造的同等性を維持することが知られていることを考慮すると、表現学習プロセスが構造的に類似したグラフに対して同様の埋め込みを生成することが保証される。

3. 提案手法

3.1 graph2vec のグラフ表現文章

graph2vec では、まずグラフを表現する文字列(本稿ではグラフ表現文章)で表現した上で、その文章に表現学習を適用する。本稿の提案手法は、graph2vec のグラフ一般の表現方法を、プログラムの構造を表す木構造に特化させたものである。そこでまず、graph2vec がグラフをどのようにグラフ表現文章で記述するかを説明する。

graph2vec は、ノード名付きグラフを対象としている。ノード名付きグラフ G_i が与えられたとき、グラフ G_i のすべてのノードを対象に、根付き部分グラフの単語表現である sg_n^d を抽出する。この時、 d は部分グラフの深さを示し、 n はノード番号を示している。この根付き部分グラフを抽出するために、WL relabeling process[7] を使用する。

個々のグラフ G を根付き部分グラフの集合 $c(G)$ と表現する。グラフを表現する文章は、式 (1) となる。

$$c(G) = \{g_{v1}^0, g_{v2}^0, \dots, g_{vn}^0, g_{v1}^1, g_{v2}^1, \dots, g_{vn}^1, g_{v1}^H, g_{v2}^H, \dots, g_{vn}^H\} \quad (1)$$

つまり、 $n(H+1)$ 個の根付き部分グラフの単語表現から構成された集合 $c(G)$ を作成する。このとき、 g_i^t は、根付き部分グラフの単語表現を表し、 i 番目のノード V_i を根とする、深さ t の部分グラフを n はグラフ内のノード数を表す。

graph2vec の流れを図 3 に示す。事前処理として、与え

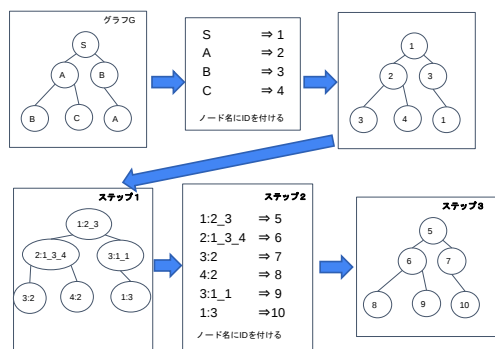


図 3 graph2vec のグラフの文章表現取得の流れ

られたグラフ G のすべてのノードに対して、ノード名にラベルを割り当てる。この時、ノード名に対するラベルを部分グラフ単語表現として抽出する。

事前処理が終了後、根付き部分グラフの単語表現取得のステップは次の 3 つである。

- (1) ノード V_i に隣接する接続ノードのラベルを集め、昇順でソートした文字列 $s_{(V_i)}^t$ を作成する。その後、 $s_{(v)}^t$ の先頭に根ノードのラベルを加え、 g_i^t を作成する。
- (2) 作成した g_i^t にラベルを割り振る
- (3) 割り振ったラベルを V_i の新しいラベルとする。また、新しいラベルを部分グラフ単語表現として抽出する。

以上の 3 ステップを与えられた回数 (H) 実行する。例として、図 3 のグラフ G の構造が与えられたとき、事前処理が終了した地点では、部分グラフの深度 0 の部分グラフ単語表現が出力される。式に表すと $c(G) = \{g_{v1}^0, g_{v2}^0, \dots, g_{vn}^0\}$ となる。具体的な値は、 $\{1, 2, 3, 4, 3, 1\}$ となる。次に、深度 1 のグラフの文章表現は、 $c(G) = \{g_{v1}^1, g_{v2}^1, \dots, g_{vn}^1, g_{v1}^2, g_{v2}^2, \dots, g_{vn}^2\}$ となり、具体的な値として、 $\{1, 2, 3, 4, 3, 1, 5, 6, 8, 9, 7, 10\}$ となる。この操作を H 回繰り返すことにより、根から深

さ H までの $n(H+1)$ 個の根付き部分グラフの集合が作成できる。

本稿では、プログラムの構造を示す木構造の分散表現を得たいと考えている。木はグラフの特殊形であるので、グラフ構造を分散表現化する graph2vec をそのまま使用することが出来るが、次の 2 点が木構造を扱う上で特化が可能な点として存在する。

1 つ目は、graph2vec がノードの階層性である親子関係を考慮していないことである。グラフ一般を扱う上で当然のことではあるが、graph2vec では親子関係を無視するため、部分グラフの深度を深くすればするほど、同じノードの周辺情報などが重複して含まれる。

具体例を図 4 に示す。図では、深度 2 の根付き部分グラフの抽出を行っている。このグラフの文章表現は、 $\{1, 2, 3, 4, 3, 1, 5, 6, 8, 9, 7, 10, 11, 12, 13, 14, 15, 16\}$ となる。ここで、ラベル 12 に含まれている、根付き部分グラフをノード番号に戻す。ラベル 12 に使われているノード番号とその関係 (根と周辺ノード) に直すと図 5 となる。

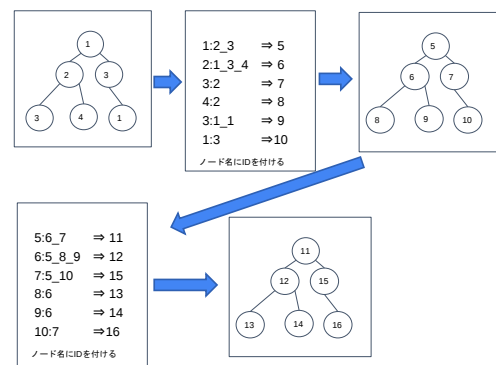


図 4 graph2vec の親子関係に対応してない問題点

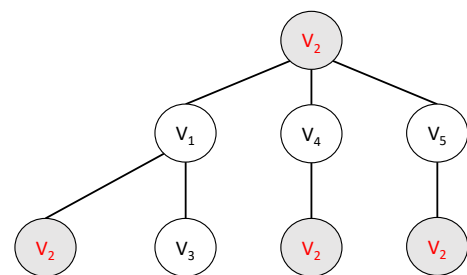


図 5 graph2vec の同じノードの参照

2 つ目は、兄弟関係を考慮しないことである。graph2vec では、ノード名にラベルを割り当て、根付き部分グラフの単語表現を取得するときには、ソートを行うため、接続ノードの順番の情報が失われる。

しかし、プログラム構造を示す AST では、文法規則を使用してプログラムを木構造に変換しており、プログラム

コード上の線形順序であるノードの兄弟関係は重要な情報となる。

具体例として、図 6 について考える。

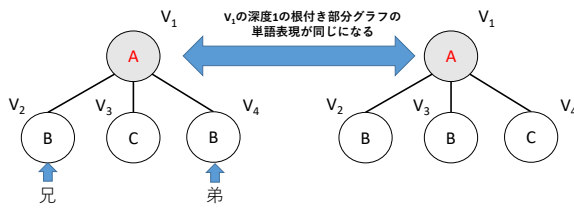


図 6 兄弟関係の問題点

2つの木にラベルを割り当てると、 $A = 1, B = 2, C = 3$ としたとき、2つの木構造の V_1 の深度1の根付き部分グラフの単語表現は $1:2.2.3$ となる。この時、兄弟関係を考慮するならば、 $1:2.3.2$ と $1:2.2.3$ の2種類の根付き部分グラフ表現にならなければならない。

3.2 提案手法の木表現文章

前節での記述を前提に、本稿では木構造に適した分散表現抽出の手法を提案する。本稿では、特定の編集操作によってコンパイルが可能なASTを作成できる木構造を分析対象とし、そのような木構造を拡張AST(以下、EAST)と呼ぶ。すなわち、プログラム作成途中のコンパイルが通らない・通常の文法による統語解析が行えないプログラムも、ASTからの編集を定義することにより、特定のASTから、特定の編集処理の過程を経た木として扱う。

例えば、main関数しか存在していないプログラムであっても、データベース上の短いコンパイル可能なプログラムのASTから削除操作の連続によって得られた編集済みの木、つまりEASTとして扱う。そのため、EASTでの編集操作にともなう等の追加定義を行い、まずASTに対する編集と、それらの編集操作と整合性を持つ形でEASTの部分木を文字列として記述可能な、ノード表現タームの表記方法を設計する。

ASTの生成を考えると、根から生成に適用された文法規則を順番に記述すれば、既に構築された木中のどのノードに文法規則の適用するかを示した系列がASTとなる。文法の適用規則は通常文字列として示すことができるので、ノードを表現する文字列(以下ノード表現タームと呼ぶ)として冗長なものとしては、ノードを生成する文法上の適用規則を文字列として記述したものが考えられる。また、木中のノードは、根からの文法規則の系列として示すことができるので、具体的には、最も冗長なノード表現タームは、根から特定ノードに至る文法規則適用の系列と、ノードを生成するために適用した文法規則を使って文字列化できる。課題はそのような冗長な文字列を以下に圧縮して記述し、木から木への編集に対応する距離を求める上で役立つ

分散表現を獲得するかにある。

ここでASTとAST、あるいはASTとEASTの近さは、ASTからの編集操作の回数の少なさで定義できる。また、EASTとEASTの近さも2つの木それぞれから最も少ない回数の編集操作で到達できる共通木を見出し、それぞれの編集操作の回数の和で定義することが可能である。本稿では、このような編集操作に基づくEAST間の類似性と妥当な相関性を持つEASTのベクトル表現を表現学習することが主眼となる。

表現学習には様々な事前の仮定が必要であるが、本稿の場合、EASTからのノード表現タームへの変換の設計を、EASTのベクトル表現の類似性から発見できる、扱うプログラム集合の持つ特徴的な構造パターンであるプログラムパターンの有用性の観点から検討する。graph2vecは、グラフ構造の根付き部分グラフを単語表現に変換した後、単語列に変換しグラフの文章として扱っている。本稿で提案するEASTからベクトル埋込の表現学習ソフトウェアを便宜上EAST2vecと呼ぶ。

EAST2vecはgraph2vecをASTに使用したときの問題点を克服できるように設計を行った。つまり、親子関係と兄弟関係を考慮したノード表現タームの作成を行っている。EAST2vecでは、 V_i のノードを根としたとき、親の情報を取得することなく、根ノード自身の情報と、その子ノードの情報でノード表現タームを作成する。また、兄弟関係を考慮するため、木の左側(長兄)から順番に記述する。このことにより、同じノード名を持った子ノードで違う兄弟関係ならば、必ず異なったノード表現タームとなる。

具体的なEASTをEAST表現文章に変換する方法を説明する。まず、事前処理として、EASTの子ノードが存在しないノードに対して'end'ノードを挿入する。これは、非終端記号を表すノードとする。その事前処理を図7に示す。次に、EAST表現文章の取得方法について解説する。大きく分けて2つのステップが存在する。

- (1) 根ノード V_i の子ノードのノード名を集め、兄から順番に羅列した文字列 $s_{(v_i)}^t$ を作成する。その後、 $s_{(v_i)}^t$ の先頭に根ノードのノード名を加え、 g_i^t を作成する。
- (2) 作成した g_i^t を V_i の新しいノード名とする。また、新しいノード名をノード表現タームとして抽出する。

4. 分散表現を利用した回答プログラム課題の判別

本論文では、グラフ構造の分散表現を獲得するgraph2vecを木構造に適した改良を行う。手法としては、EASTの木構造を文字列により表現することにより、木構造を表現する文字列を文章をベクトル空間埋め込む表現学習の手法であるDoc2vecの技術を用いて、プログラムパターンの推定に即した特徴表現モデルを獲得する。木構造の文字列化は、木構造を生成する文法を基準に、根ノードを文章の先

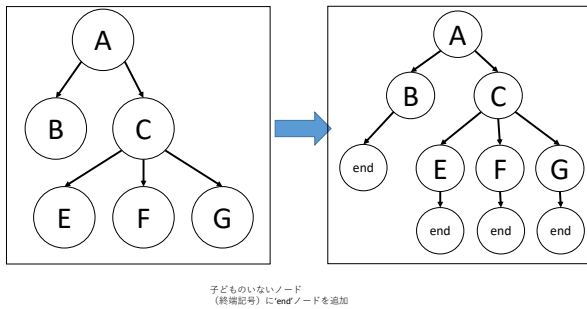


図 7 AST への事前処理

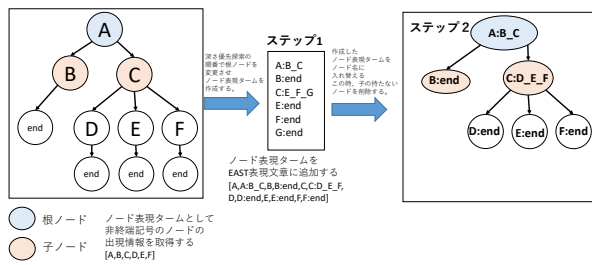


図 8 EAST 表現文章の作成の流れ

頭行と考え、文法中の規則の逐次適用の系列を文章の各行と考えた。

EAST を単語列に変換したのち、提案手法の分散表現化 (EAST2vec) により、EAST の特徴ベクトルを抽出する。この時、Doc2Vec の DBOW モデルを使用するため、EAST を単語列からなる文章に変換しなければならない。そのため、EAST を表現した文章 (以下、EAST 表現文章) を作成し、入力とする。EAST 表現文章は、EAST の部分木の文字列表現 (以下、ノード表現ターム) の集合である。EAST 表現文章を入力し、Doc2Vec の DBOW モデルで分散表現を取得する。そのため、EAST 表現文章からノード表現タームの出現を予測するモデルとなる。部分木の出現が似ている EAST の分散表現の値が近くなるため、EAST の部分木の特徴がベクトル空間へマッピングされる。

graph2vec を元に改良した EAST2vec の EAST 表現文章への変換が木構造の構造を捉えられているかを判断する。具体的には、AOJ の回答データを graph2vec と EAST2vec で分散表現を抽出し、分散表現を入力とした NN を使用し、ある回答データがどの課題に対しての回答であるかを判別する。判別精度を判定する流れを図 9 に示す。

検証は、判別スコアとして正解率を用いて行った。また、データの偏りなどを考慮するため、5 分割交差検証を行った。分割の比率は、トレーニングデータに 8 割、テストデータに 2 割とした。つまり、各課題ごとに 100 回答のプログラムが存在するため、各課題 80 個のプログラムで学習を行い、各課題 20 個のプログラムで精度の評価を行った。

この時、graph2vec と EAST2vec で使用される Doc2Vec

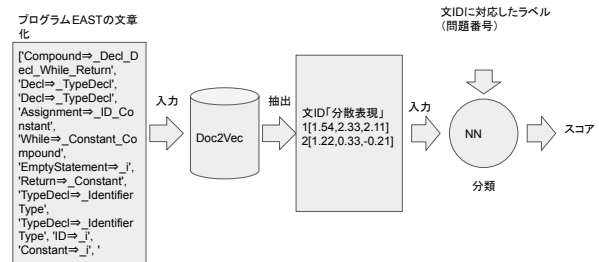


図 9 EAST の文章化から判別スコアまでの流れ

の Dbow モデルのパラメータは、200 次元の分散表現、深度 1 であり、エポック数は 300 とする。分類に使用する NN は、入力層、中間層、出力層の 3 層から構成される。パラメータは中間層のノード数は 200 とし、エポック数を 2000、バッチ数を 128 で固定した。

以上の条件での実験結果を表 1 に示す。この時、トレーニングデータは、モデルを学習するときを使用したデータのことであり、テストデータは未知のデータのことである。表は 5 分割交差検証の正解率の平均を示している。

テストデータでは、EAST2Vec は、graph2vec より 2.7 % の正解率の向上を確認できた。graph2vec でのトレーニングデータでの正解率が限りなく 1 に近いため、過学習が行われたと考えられる。この結果から、木構造の特徴をより捉えられていると考えられる。

表 1 各手法の正解率

	トレーニングデータの正解率	テストデータの正解率
提案手法	0.985	0.915
graph2vec	0.998	0.888

提案手法で分散表現を学習し、NN で分類をしたときの学習曲線を図 10 と図 11 に示す。図 10 は正解率の推移を表している。図 11 は損失関数によって導き出された損失の推移を表している。

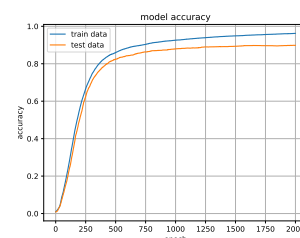


図 10 提案手法の正解率

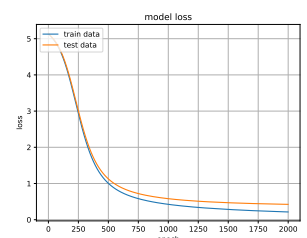


図 11 提案手法の損失の推移

graph2vec で分散表現を学習し、NN で分類をしたときの学習曲線を図 12 と図 13 に示す。図 12 は正解率の推移を表している。図 13 は損失関数によって導き出された損失の推移を表している。

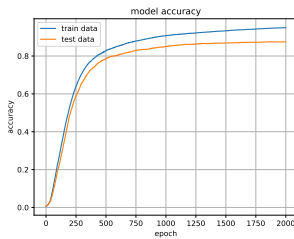


図 12 graph2vec の正解率

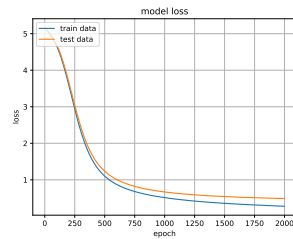


図 13 graph2vec の損失の推移

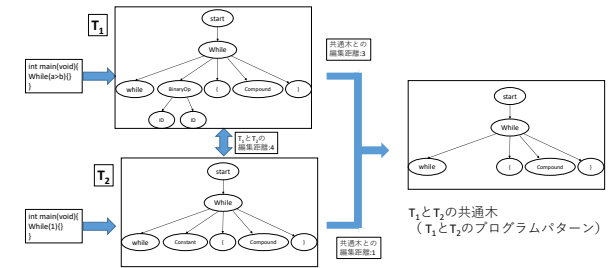


図 14 T_1 と T_2 共通部分木

5. プログラムの類型化の検討

5.1 ベクトル表現によるクラスタリングの意義

本節では、AOJ から得た、170 種類の課題の回答が記録を用いてプログラムの類型化の検討を行う。具体的には、実験データとして、各課題に対して 100 回答ずつプログラムをランダムに抽出した。このデータは、ある回答がどの課題に向けた回答であるかという <回答, 課題番号> ペアとして表現することができる。このペア集合を利用すると、任意の回答から、課題番号を推定するモデルを得ることができる。前節の実験の結果、回答プログラム課題の判別として、91.5%の精度で推定することができた。これは、本稿で提案した表現学習 (プログラム構造木のベクトル表現) が既存手法より、プログラムの特徴を表現しているといえる。

他方、本節では、EAST2Vec で得られたベクトル表現が、プログラムの類型を計算する上で適切であるかを評価するため、回答プログラムをクラスタリングすることにより、回答プログラムのクラスターを示すベクトル表現がどのようなプログラムの特徴を示すかを検討する。

この時、クラスターのプログラム特徴は、あるクラスがクラスターが木 T_1 と木 T_2 からなる時、それらの木の共通部分に基づいて定義した。木 T_1 と木 T_2 の共通部分木の抽出例を図 14 に示す。C 言語の簡単な while 文を記述しているプログラム同士である。違いは、条件式の部分である。一つは、 $a > b$ が記述されており、もう一つは '1' で記述されている。AST に変換したとき、非常に似た構造になるが、EAST 上でも条件式の部分に違いが出ている。この時、2 つのプログラムの EAST の共通木を取得したとき、条件式が記入されていない EAST が抽出できる。

つまり、提案手法によるベクトル間の類似度に基づいて得られたクラスターに含まれるプログラムの共通部分木が、そのクラスターのプログラムの特徴を説明する上で妥当であるかを検討する。また、もし妥当であるならば、それが学習上のプログラムの特徴の類型であるプログラムパターンの候補として有益ではないかと考えた。

5.2 クラスタリングの結果

前節で述べたように表現学習で得られたベクトル表現を

利用し、どの程度有益なプログラムパターンを抽出できるかを検討するために、各回答のプログラムの分散表現を Ward 法 [8][11] でクラスタリングを行った。

Ward 法は階層的クラスタリング手法の一つである。クラスタリングは、1 個の対象だけを含む N 個のクラスターがある初期状態から、クラスター間の距離関数に基づき、最も距離の近い二つのクラスターを逐次的に併合する。そして、この併合を、すべての対象が一つのクラスターに併合されるまで繰り返すことで階層構造を獲得している。なお、階層クラスタリングは最短距離法、最長距離法および群平均法は任意の対象間の距離が与えられている場合に適用できる。もし、対象が属性ベクトルで記述されている場合は、属性ベクトル間のユークリッド距離などを適用する。Ward 法の距離関数の計算式を式 2 に示す。

$$D_{wt} = \frac{1}{n_u + n_v + n_t} \{ (n_u + n_t) D_{ut}^2 + (n_v + n_t) D_{vt}^2 - n_t D_{uv}^2 \} \quad (2)$$

この階層構造は、デンドログラムによって表示される。デンドログラムは、各終端ノードが各対象を表し、併合されてきたクラスターを非終端ノードで表した 2 分木である。Ward 法は、各対象から、その対象を含むクラスターのセントロイドまでの距離の二乗の総和を最小化する。

Ward 法を利用して EAST の分散表現のクラスタリングを行った。その結果を図 15 に示す。

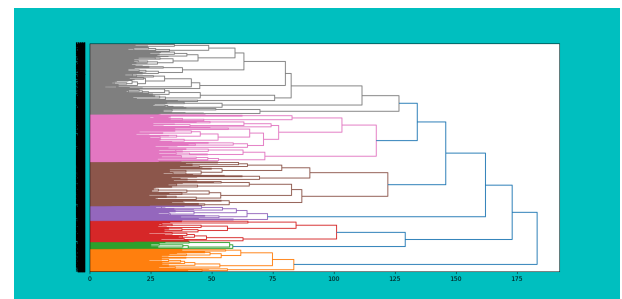


図 15 Ward 法を利用したクラスタリングの結果

EAST2vec を利用した分散表現の抽出を行い、その分散表現を利用し、階層的クラスタリングの行っている。そのため、図中の左側ほど、距離関数が少ないクラスター同士である。なお、階層的クラスタリングのある程度の課題を含

んだクラスタから、プログラムのタイプのパターンが取れる。特に、課題番号の分散の小さいクラスタは、その代表ベクトルが一定の課題に対する解答類型になっていると考えられる。

プログラムのベクトル表現で得られたクラスタに含まれるプログラム間の共通部分木を使い、プログラムの類型について検討する具体的な例を図 16 示す。

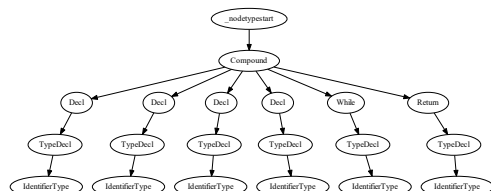


図 16 抽出した類型のプログラムパターン

この特徴木は、課題番号 ITP1.5.A と ITP1.5.B と ITP1.5.C の課題に対する回答プログラムが合成されている。課題番号と課題文と合成された回答のプログラムの数を表 5.2 に示す。

課題文から、作成されるプログラムが、入力に対応した関数宣言と繰り返し文で構成されることが分かる。これは、AOJ を元に C プログラミングを解説している文献 [9] や、アルゴリズムを文献 [10] によっても確認できる。

6. おわりに

本稿では、AOJ における 170 種類の課題に回答するプログラム事例の集合に対して AST の解析を行ったデータベースを作成した。AOJ で収集されたプログラムは目的を適切に達成しているもので、動作することは保障されており、プログラミング言語の文法に基づいて AST を得ることができる。しかし、本稿の目的は学習者の編集途中にあるプログラムを扱うことにあるため、多様な編集状態にあるプログラムも扱えるような拡張である EAST を想定してプログラムの表現学習を提案した。提案では、EAST で表現されたプログラムの木構造を文字列により表現することにより、文章をベクトル空間埋め込む表現学習の手法である Doc2vec の技術を用いて表現学習による特徴表現モデルを獲得した。

得られた特徴表現モデルの有効性を評価するため、AOJ に投稿された解答プログラムがどの課題に対して解答されたか目的変数として推定する実験を行い、91.5%の正解率を得た。このことから、提案手法で得られた特徴表現モデルの有効性を確認した。さらに、特徴表現モデルにより表現された AOJ の解答プログラムのクラスターリングを行い、得られたクラスタを代表する共通部分木が、プログラムの類型を特徴付ける上で有益なパターンであるかという観点から検討を行った。その結果、クラスターリングによって抽

表 2 結合された課題と回答数

	課題文
ITP1.5.A	たて H cm よこ W cm の長方形を書くプログラムを作成しなさい。1cm1cm の長方形を '#' で表す。入力は複数のデータセットから構成されており、各データセットの形式は以下のとおりです: H, W H, W がともに 0 のとき、入力の終わりとなる。
ITP1.5.B	以下のような、たて H cm よこ W cm の枠を描くプログラムを作成しなさい ##### #...# #...# #...# #...# ##### 上図は、たて 6 cm よこ 10 cm の枠を表している。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次のとおりである: H, W H, W がともに 0 のとき、入力の終わりとする。
ITP1.5.C	以下のような、たて H cm よこ W cm のチェック柄の長方形を描くプログラムを作成しなさい。 #.#.#.#. .#.#.#.# #.#.#.#. #.#.#.#. #.#.#.#. #.#.#.#. #.#.#.#. 上図は、たて 6 cm よこ 10 cm の長方形を表している。長方形の左上が '#' となるように描け。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 H, W がともに 0 のとき、入力の終わりとする。

出されたプログラムのクラスタからいくつかの特徴的なプログラム類型を発見することができた。

今後の課題として、本稿で述べたクラスターリングによるクラスタの検討を評価指標として、EAST のノードを表現する文字列の設計をより精緻にしていくことにより、より有益なプログラムパターン群を抽出・整理していきたい。

謝辞 本研究は JSPS 科研費 20K12101(課題名「プログラム作成・修正過程の分散表現開発と学習支援発話の生成」)の助成を受けたものです。

参考文献

- [1] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, Vol. 2, No. 3, pp. 183–192, 1989.
- [2] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, Vol. 4, No. 2, pp. 251–257, 1991.
- [3] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, Vol. abs/1405.4053, , 2014.
- [4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, Vol. abs/1301.3781, , 2013.
- [5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C.

- Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc., 2013.
- [6] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *CoRR*, Vol. abs/1707.05005, , 2017.
- [7] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, Vol. 12, No. 9, 2011.
- [8] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, Vol. 58, No. 301, pp. 236–244, 1963.
- [9] 渡部有隆. オンラインジャッジではじめる C/C++ プログラミング入門. マイナビ出版, 2014.
- [10] 渡部有隆. プログラミングコンテスト攻略のためのアルゴリズムとデータ構造. マイナビ出版, 2015.
- [11] David Wishart. 256. note: An algorithm for hierarchical classifications. *Biometrics*, Vol. 25, No. 1, pp. 165–170, 1969.