B-006

実行トレースを用いた機械学習によるソースコードの特性推定
Estimation of Source Code Characteristics by Machine Learning using Execution Traces

中里見 雄大† 山崎 憲一†
Yudai Nakasatomi Kenichi Yamazaki

1 はじめに

現在, DNN（Deep Neural Network）などの機械学習手法を用いたプログラミング支援に関する研究が盛んに行われている．例えばプログラム分類, コードクローン検出, 自動デバッグ, コメントやコードの自動生成などが挙げられるがこのようなタスクに DNN を適用するためにはプログラムの分散表現をネットワークに学習させる必要がある．既存の手法の多くは, 抽象構文木（AST）などを用いて構文情報を表現し学習する手法や, ソースコードを文章として捉えトークン列を学習する手法といったプログラムの静的な解釈に基づくものである．しかしプログラムはソースコード上のごくわずかな違いがその挙動に大きな違いをもたらす場合がある．そこでプログラムの持つ動的な情報を学習に利用すればより良い分散表現の獲得が可能となり, 様々な下流タスクに応用できると考えられる．本研究では動的な情報として, プログラム内の変数集合の値の時間的推移（実行トレース）を利用する．

また近年, プログラムの実行トレースと同じく時系列データを扱う自然言語処理（NLP）では, 従来の RNN（Recurrent Neural Network）をベースとした Seq2Seq モデルに代わって, Self-Attention を用いた Transformer モデルが大きな成果を上げている．Transformer モデルは RNN のように再帰処理を行わず並列計算が容易でありモデルの学習にかかる時間が削減できるため, NLP においてのデファクトスタンダードとなりつつある．

そこで本稿では, Transformer モデルでプログラムの実行トレースを扱うためのベクトル化方式を 3 つ提案し, 下流のタスクとしてソースコードからプログラム特性（メモリ使用量もしくは CPU 実行時間）を予測するモデルについて述べる．

2 関連研究

プログラムの実行トレースを DNN の入力として扱った研究について述べる．文献 [1] ではプログラムの実行トレースを入力として複数の GRU（Gated Recurrent Unit）を用いてプログラムのバグの種類の推定を行った．ここではプログラム中の変数の値を一度全て辞書に登録し, その番号を GRU の入力として学習を行っている．しかしこの手法では実行トレース中に様々な数値が出現する場合の学習は困難であると考えられ, 扱うことが出来るプログラムが強く限定されるという問題がある．

文献 [2] では, バグを含んだ実行トレースから Seq2Seq モデルを用いて正しい実行トレースを生成することを試みている．この研究では実行トレースを固定長ベクトルとして Seq2Seq モデルで扱うために Padding トークンと変数値, 変数の数, ステップ数の各次元の終わりを示す 3 つの終端トークンを組み合わせた 4 つのベクトル表現方式を提案した．評価の結果, 入力ベクトルの 1 次元への平坦化と Padding を組み合わせた方式が最も高性能であったが, いずれの手法も下流タスクにおける推定の精度は低かった．

† 芝浦工業大学大学院
Shibaura Institute of Technology graduate school

3 提案手法

Transformer モデルで実行トレースを扱うための 3 つのベクトル表現方式を提案する．本研究では C 言語で記述されたプログラムを対象とし, それをステップ（1 行ずつ）実行した際に各ステップで得られる変数の値の時間的シーケンスを実行トレースとする．処理が合計 n ステップのプログラム P と変数の集合 $V(v_0, v_1, ..., v_{m-1} \in V)$ が与えられた時, あるステップ t における変数 v_1 の値を $x_{t\_v_1}$ とする．ここで x は次のような方法で得る．変数の値をテキスト表現した後に, NLP で単語分散表現の獲得に用いられる fastText によって内部表現ベクトルに変換する．

3.1 提案１：変数値トレース

プログラム中の m 個の変数それぞれをステップごとの値のシーケンスとして表現し, その集合を変数値トレースとする．この方式のモデルへの入力 $\mathbf{X}$ は式 (1) で示されるような n行 × m列 のテンソルとなる．

$$v_0 = (x_{t\_v_0}, x_{t+1\_v_0}, x_{t+2\_v_0}, \dots, x_{n\_v_0})$$

$$v_1 = (x_{t\_v_1}, x_{t+1\_v_1}, x_{t+2\_v_1}, \dots, x_{n\_v_1})$$

$$\dots$$

$$v_{m-1} = (x_{t\_v_{m-1}}, x_{t+1\_v_{m-1}}, x_{t+2\_v_{m-1}}, \dots, x_{n\_v_{m-1}})$$

$$\mathbf{X} = [v_0, v_1, v_2, \dots, v_{m-1}] \quad (1)$$

この方式ではプログラム中の変数の数に応じて入力のサイズが大きくなるという問題がある．そこでプログラム実行中に値の変化が少ない変数については, 入力から除外することで入力のサイズを削減する．

3.2 提案２：ステップトレース

プログラム中の各ステップをそのステップ中の m 個全ての変数の値で表現し, そのシーケンスをステップトレースとする．この方式のモデルへの入力 $\mathbf{X}$ は式 (2) で示されるような m行 × n列 のテンソルとなる．

$$s_1 = (x_{t\_v_0}, x_{t\_v_1}, x_{t\_v_2}, \dots, x_{n\_v_{m-1}})$$

$$s_2 = (x_{t+1\_v_0}, x_{t+1\_v_1}, x_{t+1\_v_2}, \dots, x_{t+1\_v_{m-1}})$$

$$\dots$$

$$s_n = (x_{n\_v_0}, x_{n\_v_1}, x_{n\_v_2}, \dots, x_{n\_v_{m-1}})$$

$$\mathbf{X} = [s_1, s_2, s_3, \dots, s_n] \quad (2)$$

この方式ではプログラムの総ステップ数に応じて入力のサイズが大きくなってしまう．そこでプログラム実行中に

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

変数の数値変化が少ないステップ区間については，入力から除外することで入力のサイズを削減する．

3.3 提案３：変数間自己注意ステップトレース

変数値トレースやステップトレースでは変数間の関係を捉えることが難しい可能性がある．また文献 [1] でも変数間の依存関係を考慮したモデルが最も高性能であった．そこでモデルに新たな Self-Attention 層を追加し，式 (3) で示すように提案 2 の各ステップの変数間で Self-Attention 処理を行う変数間自己注意ステップトレースを提案する．これによって変数間の関係を考慮しながら各ステップの学習を行うことが可能と考えられる．ここで d は Q と K の次元であり，Q, K, V は提案 2 の s_n の線形変換で得られる．各重み W^Q, W^K, W^T はこの方式でモデルに追加される学習パラメータである．

$$
\begin{aligned}
Q = s_n \mathbf{W}^Q,\ K = s_n \mathbf{W}^K,\ V = s_n \mathbf{W}^V \\
s'_n = Attention(Q, K, V) \\
= softmax(\frac{Q^T K}{\sqrt{d}})V
\end{aligned} \tag{3}
$$

$$
\mathbf{Y} = [s'_1, s'_2, s'_3, \ldots, s'_n] \tag{4}
$$

モデルへの入力は提案 2 のステップトレースと同様であり，前述の Self-Attention 層の出力 $\mathbf{Y}$ は式 (4) で示される．この方式では提案 1, 2 のような入力サイズの削減は行わない．

4 実装

ここでは本研究で用いるデータセットの詳細と実行トレースの取得方法，学習モデルのアーキテクチャについて述べる．

4.1 データセット

学習データセットは IBM によって公開されたオープンソースの大規模データセットである CodeNet を利用して生成する．CodeNet は AtCoder と AIZU Online Judge という 2 つのプログラミングコンテストに実際に提出された約 1400 万件のソースコードとその問題や提出物に関するメタデータから構成されている．提出されたソースコードは C 言語，C++, Python, Java など 55 種類の言語で記述されており，メタデータはそのプログラムの特性（メモリ使用量，CPU 実行時間，コードサイズ，エラーの種類など）である．本研究では C 言語で記述されたソースコードを対象とし，メモリ使用量もしくは CPU 実行時間を推定する．そこでバグがないとタグ付けされたプログラムを付属のテストケースを用いて実行し，実行トレースのデータセットを生成する．

4.2 実行トレースの取得方法

実行トレースは文献 [2] と同様に GNU デバッガを用いて 1 ステップごとの全ての変数値をテキスト表現で取得する．本研究で扱うプログラムのほとんどがメイン関数のみで構成されているため，取得対象はメイン関数内のローカル変数のみとし，グローバル変数（外部変数）や呼び出した先の関数（ユーザ定義の関数だけでなく，標準入出力などのライブラリ関数も含む）内の変数は扱わない．配列は変数の集合として扱い，各ステップで配列の各要素の値を全て取得する．また構造体やポインタなどの C 言語の特殊なデータ型は考慮しない．

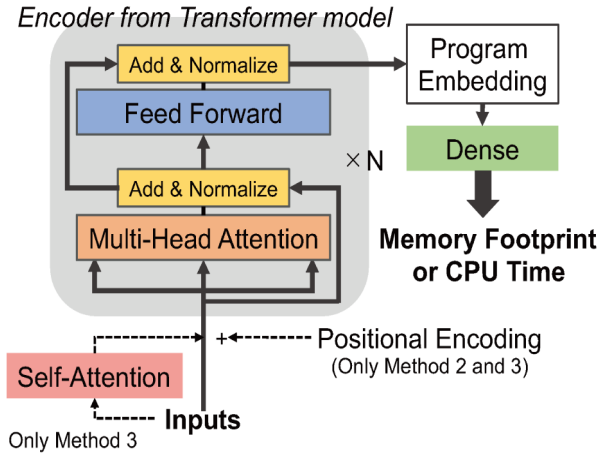

図 1 モデルのアーキテクチャ

4.3 モデルのアーキテクチャ

モデル全体のアーキテクチャを図 1 に示す．このモデルではまず Transformer モデルのエンコーダーを用いて実行トレースからプログラムの分散表現である Program Embedding を得る．次にそれを Dense 層へ入力しメモリ使用量や CPU 実行時間の予測を出力する．

Transformer モデルのエンコーダーは Multi-Head Attention 層，残差結合およびレイヤー正規化層，Feed Forward 層から構成され，これらの処理を複数回繰り返す．提案 3 の方式でのみ入力の直前に Self-Attention 層を追加する．Transformer モデルで用いられる Positional Encoding は，入力データに順序情報を付与するための仕組みである．ここで提案 1 の変数値トレースは変数を表したベクトルの集合であり，変数間の順序関係は予測に影響しないと考えられるため順序情報の付与は行わない．提案 2 のステップトレースおよび提案 3 の変数間自己注意ステップトレースは各ステップを表わすベクトルのシーケンスであるため Positional Encoding を用いて入力に順序情報を付与する．

一般に Transformer モデルの学習においても通常の DNN と同じく入力のベクトル長を固定する必要がある．そこで本研究では，データセットにおける変数の最大個数，プログラムの最大ステップ数に合わせて Padding トークンを追加することで入力のベクトル長を固定して学習を行う．

5 おわりに

本稿では，Transforme モデルでプログラム実行トレースを扱うための 3 つのベクトル化方式を提案し，ソースコードからプログラム特性としてメモリ使用量もしくは CPU 実行時間を予測するモデルについて述べた．今後はモデルの実装及び評価を行う予定である．

参考文献

[1]Ke Wang, Zhendong Su, and Rishabh Singh. ”Dynamic neural program embeddings for program repair”, International Conference on Learning Representations, 2018.

[2]Takuma Koyama and Kenichi Yamazaki. ”Multi-Dimensional Vector Representation of Execution Trace for Automatic Debugging”,SEATUC poster session, 2020.

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.