機械学習を用いたコード内フォールト自動修正に関する研究

M2018SE002 青木公宏
指導教員：野呂昌満

1　はじめに

機械学習を用いたプログラムの修正の研究がされている．フォールトを含むプログラムコードと修正されたプログラムコードから，修正方法や正しいプログラムコードの構造を学習することで，自動的に修正を行う．

コード内フォールトは，以下の 4 つに分類できる．

- 構文フォールト
- 静的意味フォールト
- 動的意味フォールト
- 論理フォールト

機械学習によるフォールトの発見・修正は，対象とするフォールトによって何を入力とするかが決まる．

Gupta ら [3] の DeepFix は，字句列を入力とし，コンパイラによって検出される構文フォールトと静的意味フォールトを修正の対象としている．Wang ら [1] は，変数と値の組である環境トレース情報を入力とするニューラルネットワークを設計し，動的意味フォールトと論理フォールトに関する発見・修正が行えることを示した．

本研究の目的は，動的意味および論理フォールトの一部の自動修正である．CDI ツールの発想を手掛かりに，全ての変数の定義参照トレースである環境の定義参照トレースと制御構造を入力とする．制御フローから得られる情報を入力とするので，実行前に検査が可能であり，全てのパスを網羅的に解析を行う．

本研究の研究課題は以下の 3 つである．

1. 環境の定義参照トレースを入力とする制御フローに着目したニューラルネットワークの設計
2. エンコーダーデコーダーモデルを対象としたニューラルネットワークとの共調
3. GNN(graph neural network) とエンコーダーデコーダーモデルの共調に関する考察

本研究では，変数の def-use を解析対象とするので，変数の環境が変わるごとに変数の定義と参照を抽出する．抽出した定義参照トレースとプログラムの制御構造をもとに，動的意味フォールトと論理フォールトの一部を対象とするニューラルネットワークを設計する．

修正を行うために，字句列を入力として修正を行うエンコーダーデコーダーモデルと設計したニューラルネットワークを共調させる．字句に関する特徴と変数の def-use に関する特徴を対象とした解析を行うことで，変数の def-use を考慮した自動修正が実現できると考える．

本研究で設計したニューラルネットワークは，一つのパスについて解析を行うネットワークであったので，GNN に拡張し，字句列を入力とするエンコーダーデコーダーモデルと共調させることで自動修正を実現する．

2　背景技術

2.1　Gated Graph Neural Network

GGNN はグラフを扱うニューラルネットワークである．グラフは $G = (V, E, X)$ の三つ組で構成される [2]．V はノードのセット，X はノードの特性，および全ての有向エッジのセット $E = (E_1, \ldots, E_K)$ で構成される．K はエッジタイプの種類である．

全てのノード v を，ノードラベル $x(v)$ から初期化された状態ベクトル $h(v)$ に関連付け，グラフ全体に情報を伝播するために，タイプ k のメッセージが各 v から近隣のノードに送信される．各メッセージは現在の状態ベクトルから $m_k^{(v)} = f_k(h^{(v)})$ として計算される．

すべてのグラフエッジのメッセージを同時に計算することで，全ての状態を同時に交信が可能．

3　関連研究

3.1　DeepFix

Gupta ら [3] は，コンパイルエラーに起因するフォールトを対象にした自動修正の学習モデルである DeepFix を設計した．フォールトを含むソースコードとそれを修正したソースコードのペアから修正方法を学習する．このペアをトレーニングデータとして扱い，修正方法と各字句の関係を学習することで，新規のコード内フォールトを含むソースコードが与えられた際に適切な修正を行う．

ソースコード全体を分析し，推論を行うために Attention 付き Encoder-Decoder モデル (図 1) を用いて設計された．

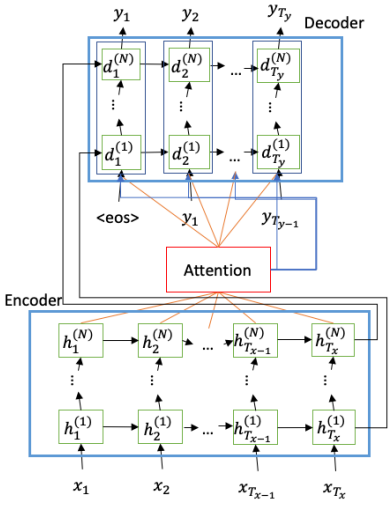

図 1　Attention 付き Encoder-Decoder モデル

入力系列は Encoder で処理され，入力系列の処理が全て終わった後に Decoder が出力系列の生成を開始する．

Attention 機構は出力を生成する際に，任意の時刻のデコーダーの隠れ状態と各時刻のエンコーダーの隠れ状態から計算し，適切な入力情報を選択する．

3.2 Wang らの研究

Wang らはプログラムの意味上の表現をニューラルネットワーク上で活用するために，プログラムの実行トレースからセマンティクスを解析する方法として，State Trace Model を提案した．変数と値の組である環境トレース情報を入力とし，動的意味フォールトと論理フォールトを対象とする．図 2 に State Trace Model を用いた状態トレース埋め込みのワークフロー全体を示す．

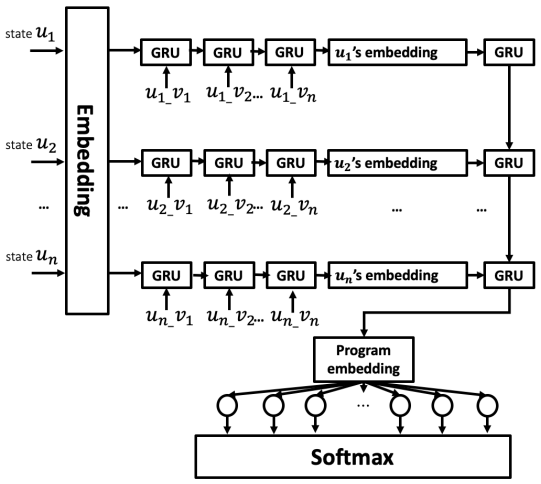

図 2 State Trace Model のワークフロー

変数値の連続タプルとして表されたプログラム状態を RNN を用いてエンコードし，全てのプログラム状態埋め込みをシーケンスとして別の RNN の入力として扱う．プログラム全体のセマンティクスに関する特徴をエンコードし，RNN の最終状態を入力とする出力層により分類を行う．

4 設計

4.1 設計指針

設計指針を以下に定義する．

- 変数の定義参照系列をもとにフォールトの有無ならびに分類を行う
- エンコーダーデコーダーモデルとの共調を可能とする
- GGNN への拡張を考慮する

変数の定義参照系列をもとにフォールトの有無ならびに分類を行う．環境の定義参照トレースと制御構造を入力とし，動的意味フォールトと論理フォールトの一部を対象にしたニューラルネットワークを設計する．

得られた制御フローに関する特徴を，字句列を対象とした自動修正を行うニューラルネットワークとの共調を可能とする．エンコーダーデコーダーモデルと共調させることで，字句列と環境の定義参照トレースから解析できる特徴をエンコードし，修正を実現する．

GGNN を用いて解析を行うことで，プログラムの構文に沿いながらパスを考慮した環境のエンコードが行える．字句列を入力とするエンコーダーデコーダーモデルと共調することで，適切な修正を行う．

4.2 変数の定義参照系列をもとにフォールトの有無ならびに分類を行うネットワークの設計

ニューラルネットワークモデルを設計する上で，以下の 2 つのことに考慮する必要がある．

- 各箇所での変数の定義参照系列をエンコードする方法
- プログラム全体の環境の定義参照トレースを解析する方法

変数の定義と参照で表すプログラムの環境は，代入文や条件式で変わるので，それぞれの箇所ごとにエンコードする必要がある．

プログラム全体について分析するために，エンコードされた環境ベクトルを入力とするような解析が必要であると考えられる．

以上のことに考慮し，本研究で設計するニューラルネットワークモデルの設計指針を次のように定義する．

- Wang らの State Trace Model[1] の拡張としてニューラルネットワークモデルを設計
- カテゴリカル属性を入力として扱うニューラルネットワークモデルを設計

本研究では，それぞれの箇所ごとに変数の定義参照系列をエンコードし，プログラム全体の制御フローについて分析するために，State Trace Model をもとにニューラルネットワークモデルを設計する．

State Trace Model では，入力に数値属性を扱っていたが，本研究で入力として扱うのはカテゴリカル属性である．

図 3 に本研究で設計したニューラルネットワークモデルを用いたワークフローを示す．

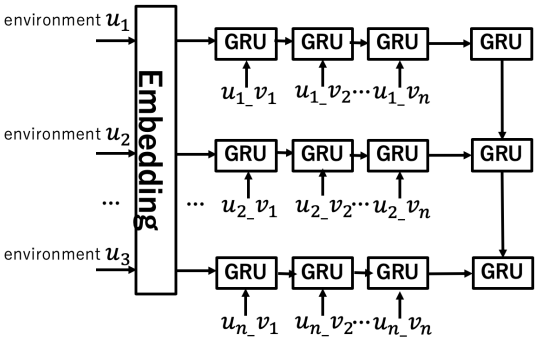

図 3 変数の def-use をエンコードするニューラルネットワーク

箇所ごとに変数の定義参照系列をエンコードし，エンコードされた全ての環境ベクトルを GRU で分析することで予測をする．

入力となる変数は，定義・参照・未出現の 3 つに分類できると仮定する．変数は Embedding 層で分散表現に埋め込まれ，GRU に与えられる．各箇所での GRU は，各変数の並びとその分類を特徴として抽出し，隠れ状態に保持する．エンコードされたプログラムの環境ベクトル全てに対して GRU を用いて分析を行い，各箇所でエンコードされた変数の並びと分類から，プログラム全体の変数の定義・参照に関する特徴を得る．

4.3 エンコーダーデコーダーモデルとの共調

前述したネットワークで解析したプログラムの環境を，エンコーダーデコーダーモデルと共調させ，自動修正を行う方法を提案する．

図 4 に示すように，エンコーダーデコーダーモデルに字句を与えるさいに，その字句が読まれる時のエンコードされたプログラムの環境を単語ベクトルと同時に入力する．

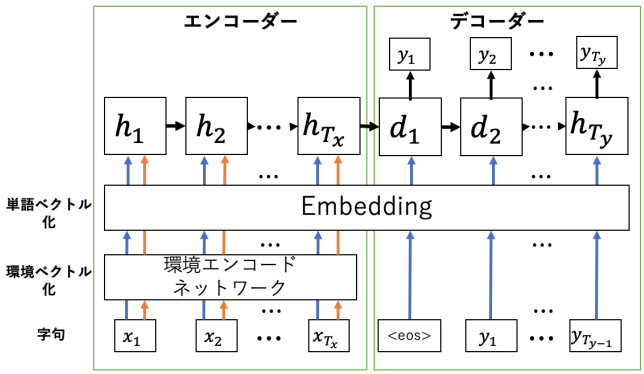

図 4 エンコーダーデコーダーモデルとの共調

エンコーダーデコーダーモデルの入力に，字句列と環境の定義参照トレースと制御構造を扱うことで，動的意味フォールトと論理フォールトの一部の修正を目指す．

4.4 GGNN への拡張

エンコーダーデコーダーモデルと本研究で設計したニューラルネットワークモデルを用いた解析を実現するために，GGNN を適用し，エンコーダーデコーダーモデルと連携する方法を示す．

字句と同時に環境ベクトルをエンコーダーデコーダーモデルに入力するために，構文図に基づいた GGNN を設計する．字句が与えられたさいに，GGNN の対応するノードに変数の定義参照系列をエンコードする．直前までのノードの状態を伝播させることで，GGNN のノードは環境ベクトルをエンコードする．これにより，図 5 のように，エンコーダーデコーダーモデルに字句を入力するのと同時に，字句に対応した環境ベクトルを入力することができる．

5 実験

5.1 概要

if・wile のみで構成される C ソースコード 359 個を対象に，設計したニューラルネットワークを用いてフォールトを分類する実験を行なった．実験では，以下の 4 つのフォールトを対象とした．

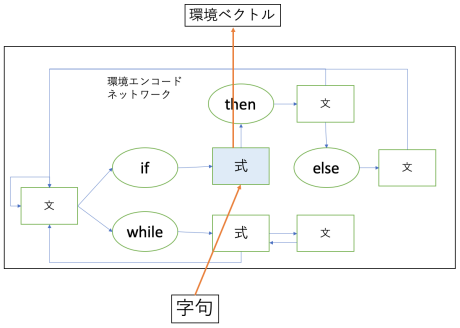

図 5 GGNN の適用

- 無限ループ
- 実行されないパスが存在
- null ポインタ参照
- 未定義変数の参照

プログラムの環境の定義参照トレースと制御構造を抽出し，入力データとした．プログラムには 1 種類のフォールトしか含まれないとし，プログラムごとに正解ラベルを与える．

5.2 システム構成

設計したシステムの概要を図 6 に示す．

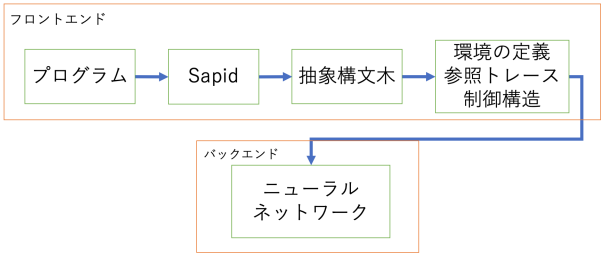

図 6 システム構成

設計したシステムは，プログラムコードから入力データを生成するフロントエンドと，学習または推論を行うバックエンドにより構成される．C ソースコードのための CASE ツール・プラットフォームである Sapid[5] を用いて抽象構文木を生成し，パスごとに走査することで環境の定義参照トレースと制御構造を抽出する．抽出した入力データをバックエンドに与え，ニューラルネットワークが学習または推論を行う．

5.3 学習モデル

学習モデルは Python を用いて実装した．ニューラルネットワークは，8 つのノードを内部に持つ 3 つの GRU で構成される．1 つ目の GRU ではそれぞれの箇所の変数の定義参照系列をエンコードし，2 つ目の GRU で各パスの環境ベクトルをエンコードする．3 つ目の GRU では各パスの環境ベクトルからプログラム全体を解析する．

入力は（定義，参照，未出現，if，while）の One-hot ベクトルとし，Embedding 層を用いて 5 次元の分散表現に変換する．それぞれの箇所ごとに，変数が定義，参照，未出現のどれに分類されるかを One-hot ベクトルで表す．制御構造を表現するときには，全ての変数を特定の制御構造を表した One-hot ベクトルにし，その箇所の入力とする．

出力層は（フォールトなし，無限ループ，実行されないパスが存在，その他）にラベル付けされた 4 つのノードから構成され，softmax 関数を用いて出力される．その他は null ポインタ参照と未定義変数の参照である．

学習の詳細を表 1 に示す．

表 1 学習の詳細

学習アルゴリズム	Adam
損失関数	クロスエントロピー
評価関数	正解率
エポック数	10

5.4 結果

実験を行なった結果，正解率は上昇していたが，損失関数は一定であり学習が収束しておらず，失敗であった．

6 考察

6.1 実験結果についての考察

実験の結果から，学習が収束しなかった原因について考察する．学習が収束しなかった原因には，一つのプログラムの全てのパスは同じラベルを付けられることから，どれがフォールトを含むパスなのかモデルが学習をしきれなかったと考えられる．図 7 を例にあげるように，設計したモデルではプログラムの複数のパスに対して同じラベルが付けられる．フォールトを含まないパスに対しても含むパスとして学習を行ってしまったことが，学習が収束しなかった原因と考えられる．

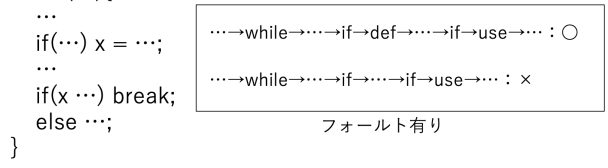

図 7 複数のパスを同時に学習するさいの問題

6.2 共調に関する考察

本研究で設計したニューラルネットワークとエンコーダーデコーダーモデルを共調させることについての考察を行う．

本研究で設計したニューラルネットワークは，それぞれの箇所ごとに変数の定義参照系列を GRU を用いてエンコードし，最終状態を入力とする GRU を用いて環境ベクトルをエンコードする．それぞれの箇所の GRU の出力がその箇所での環境ベクトルとなる．エンコーダーデコーダーモデルに，単語ベクトルと対応する環境ベクトルを与え，1 つのベクトルにエンコードする．抽出したベクトルをデコードすることで，動的意味フォールトと論理フォールトの一部の修正が実現できると考察する．

6.3 GNN を用いたニューラルネットワークの考察

GNN を用いて設計したニューラルネットワークの改善について考察する．

制御フローグラフに基づいた GNN を設計することで，学習の問題であった複数のパスを独立に扱うことを改善する．制御フローグラフはノードと有効エッジによりプログラムの全てのパスを表すので，GNN に用いることで制御構造を学習の対象とすることができる．RNN を用いて変数の定義参照系列をエンコードし，各ノードを初期化する．ノードの状態を伝播させることで，各ノードに環境ベクトルをエンコードする．全てのパスを一つのニューラルネットワークで扱うことで，学習を改善することが可能と考える．

7 おわりに

本研究では，制御フローに関するフォールトを対象に修正を行うニューラルネットワークモデルを設計した．変数の def-use に関する問題を扱い，変数の定義・参照を入力として与えるニューラルネットワークを設計することで，プログラムの環境を特徴ベクトルとして扱えるようにした．エンコーダーデコーダーモデルと共調させ，GGNN を用いることで変数の def-use に関する修正が行えるネットワークを設計した．設計したニューラルネットワークモデルを用いてフォールトの分類の実験を行い，その結果から改善方法を考察した．

今後の課題として，GNN を用いてネットワークを設計し，実装して妥当性の確認を行うことが考えられる．

参考文献

[1] K. Wang, R. Singh, Z. Su, “Dynamic Neural Program Embeddings For Program Repair,” *International Conference on Learning Representations*, 2018.

[2] L. Yujia, T. Daniel, B. Marc, Z. Richard, “Gated graph sequence neural networks.” *International Conference on Learning Representations*, 2016.

[3] R. Gupta, S. Pal, A. Kanade, and S. Shevade, “DeepFix: Fixing Common C Language Errors by Deep Learning,” *Thirty-First AAAI Conference on Artificial Intelligence*, pp. 1341-1351, 2017.

[4] 手塚太郎，しくみがわかる深層学習，朝倉書店，2018.

[5] 福安直樹，山本晋一郎，阿草清滋，“細粒度ソフトウェア・リポジトリに基づいた CASE ツール・プラットフォーム Sapid，”情報処理学会論文誌，vol.39，no.6，pp.1990-1998，1998.