JSiSE Research Report
vol.37,no.6(2023-3)

ソースコード処理手順の学習によるコード補完機能の検討

齋藤 愛莉佳[*1], 鷹野 孝典[*2]
*1 神奈川工科大学大学院 工学研究科 情報工学専攻
*2 神奈川工科大学 情報学部 情報工学科

A Method for Code Completion Function by Learning a Sequence of Procedures in Source Code

Erika Saito[*1], Kosuke Takano[*2]
*1 Course of Information and Computer Sciences, Graduate School of Engineering, Kanagawa Institute of Technology
*2 Department of Information and Computer Sciences, Faculty of Information Technology, Kanagawa Institute of Technology

For the efficient program coding, it is important to provide learning environments for improving programming skills and tools to support software development. This study presents a method to perform code completion according to the distance of meaning between procedures in a source code by learning the meanings that occur in the context of the processing procedures. In the experiment, by applying word2vec, which learns semantic relationships between words, to a logic corpus generated from a large set of source code, we evaluate the performance of code completion using the feature representation matrix of the code procedure obtained through the training with word2vec, and confirm the feasibility of the proposed method.

キーワード: プログラミング学習，ソースコード推薦，word2vec，特徴表現行列

1. はじめに

ソフトウェア開発者がプログラミングを効率的に行うために，プログラミング技能を向上させるための学習環境やソフトウェア開発を支援するツールの提供が重要である．

我々はこれまで，ソフトウェア開発者がコードを作成中に利用可能なソースコード推薦を行うことを目的として，ロジック・コーパスと呼ばれる，ソースコード集合を機能ラベル列集合として表現したコーパスを生成し，ニューラルネットワークモデル（以降，NN モデル）へ適用することで，ソースコードの処理手順の文脈を考慮したコード機能の補完手法を提案してきた[2]．しかし，この手法では，ソースコードの処理手順に着目して学習を行う系列変換型の NN モデルを利用した推定を行っており，処理手順の前後関係から生じる処理手順間の意味的関係には着目していなかった．

本研究では，ソースコードにおける処理手順に生じる意味を学習することにより，処理手順間の意味の近さに応じてコード補完を行う手法を提案する．実験では，大規模なソースコード集合から生成したロジック・コーパスを単語間の意味的関係を学習する NN モデルの一つである word2vec[6] に適用し，学習により得られたコード処理手順の特徴表現行列を利用したコード補完機能が実現可能であることを確認する．

2. 関連研究

従来研究において，NN モデルを用いたソースコード分析性能の向上もあり，ソースコード生成や推薦手法の研究が活発になっている．

文献(1)において，Li らは大規模なサンプリングとフ

ィルタリングを組み合わせ，深い推論を必要とする問題に対して新しい解を生み出すことができるコード生成システム AlphaCode を提案した．AlphaCode は，競技プログラミングコンテストにおいて上位 54%以内のとなる性能を達成している．

また，内山らは，word2vec[6]の学習基準である近傍単語の個数を増やすことにより，ソースコード特有の影響を考慮した類似コード片の推薦手法を提案した[2]．Ye らは 2 つのコアコンポーネントで構成される，エンドツーエンドの推論コード類似システムを提案した[3]．このシステムの特徴として，ソースコードの構文から意味構造の抽出機能，およびニューラルネットワークを用いたコード類似度算出アルゴリズムが挙げられる．なお，評価実験では，code2vec[7]，code2seq，neural code comprehension，aroma[10]などのコード類似度システムよりも精度が改善されたとする結果を示している．山本は，制御文の有無情報を追加したソースコード・コーパスを用いたコード補完方式を提案し[8]，さらに，文献(9)において，既存のソースコードに記述されているメソッド呼び出し文の順序に着目した，メソッド呼び出し文を補完する方式を提案した．なお，回帰結合ニューラルネットワーク(RNN，Recurrent Neural Network)を適用することで，次に現れるであろうメソッド呼び出し文を予測している．

文献(10)において，Barnaby らは機械学習を用いた，大きなコードベースから洞察を得るプロセスを大いに簡単にするコード間検索及び推奨ツールを提案した．このツールは，コードスニペットを使用して検索クエリを実行した後，コードスニペットのクラスターを検出することで，手動で数十個のコードスニペットを使用することなく，一般的なコーディングパターンを見付けられるようユーザにソースコードを推薦する．

3. 提案手法

3.1 ロジック・コーパス

提案手法では，ソースコード機能分析により，ソースコードの処理手順をコード片に対応する機能ラベル列(表 1)として抽出し，ソースコード集合を機能ラベル列集合として表現したロジック・コーパス(表 2)を生成する(図 2)．ここで，機能ラベルの抽出には code2vec などの既存手法を用いることができる．次に，NN モデルを適用したロジック・コーパスの深層学習により，ソースコードの機能を推定・補完するコード機能推定モデルを構築する．

ソースコードの処理手順を推定するために，時系列を対象とした深層学習モデルである，LSTM や双方向 LSTM(Bi-LSTM, Bidirectional LSTM)を適用する．機能ラベル列の続きとなる機能を推定する場合は，機能ラベル列をコード機能推定モデルに入力し，出力として続く機能を推定する．また，機能ラベル列を補完する機能を推定する場合は，補完する機能より前の機能ラベル列を順方向で入力し，かつ補完する機能より後の機能ラベル列を逆方向で入力することにより，それぞれの出力で得られた機能の双方を満たすものを補完機能として推定する．

表 1　機能ラベルの例

['main']
['get', 'key']
['operates', 'on', 'fact', 'handles']
['cancel', 'button', 'action', 'performed']

表 2 ロジック・コーパスの例

['test', 'lock', 'timeout'] ['sleep']
['main'] ['do', 'work']
['set', 'message', 'body'] ['load'] ['set', 'subject'] ['process']
['run'] ['get', 'value']

3.2 系列変換モデルへの適用

図 3 は，Transformer や LSTM(Long Short-Term Memory)などの系列変換 NN モデルを適用したコード機能推定モデルを構築するためのロジック・コーパスの作成手順および学習手順を示している．

図 3 に示すように，あるソースコードを l 個のコード片に分割し，それに対応する機能ラベル列 $K=\{k_1, k_2, \ldots, k_{l-1}, k_l\}$ を生成し，ソースコード集合から得られる機能ラベル列集合を，ロジック・コーパスとして生成する．また，学習データを生成する際は，ロジック・コーパスからソースコード中のコード片に対応する機能ラベル列 $K=\{k_1, k_2, \ldots, k_{l-1}, k_l\}$ を抽出し，各機能ラベル列 K において，先頭から l-1 個目までを入力データ K_{input}，末尾の l 個目を正解の機能ラベル K_{label} として作成す

る．入力データ K_{input} と正解の機能ラベル K_{label} のペアを学習データとして，系列変換 NN モデルを適用した学習を行う．

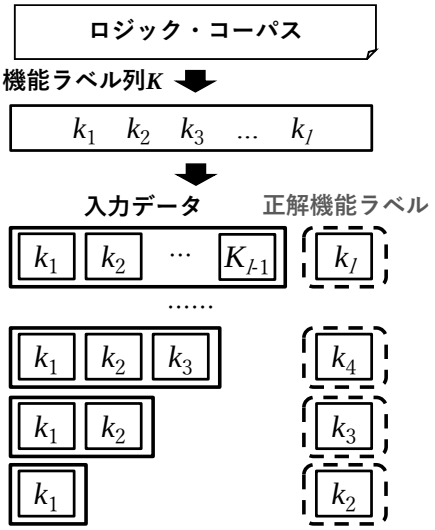

図 1 系列変換 NN モデルの学習データの生成

3.3 意味抽出モデルへの適用

ロジック・コーパスを word2vec や BERT などの単語や文章の意味抽出が可能な NN モデルに適用させ，ソースコードの補完機能の推定へと応用する．本節では，word2vec への適用方法について述べる．word2vec は，単語の意味的な特徴を多次元の単語ベクトルとして表現することで，単語ベクトル同士で演算することができる．

word2vec への学習データとして，ロジック・コーパスをそのまま適用できる．word2vec の学習アルゴリズムである CBOW (Continuous Bag-Of-Words)や skip-gram (Continuous Skip-Gram)を適用することで，コード処理手順について穴埋め箇所を生成し，処理手順の前後関係から意味を推定できるように学習する．

このように word2vec は，意味的な特徴表現を学習するため，与えたソースコードの処理手順に対して，コード機能の意味的な類似性を学習することができる．学習結果として得られるコード機能の特徴表現行列を用いて，コード補完に応用できる．入力コンテキストとしては，与えたいコンテキストおよび除外したいコンテキストを，それぞれ複数のコード機能として指定することで，ソースコードの処理手順の文脈に応じたコード機能を推定し，コード補完に利用する．

4. 実験

提案手法を word2vec に適用してコード補完モデルを構築し，モデルの推定結果からコード補完が可能であることを確認する．

4.1 実験環境

実験で使用したデータの内訳を表 1 に示す．ロジック・コーパスの生成には，Project CodeNet[11] というデータセットおよび書籍[12]に収録されているソースコードを用いた．これらのデータについて，2～10 個の機能ラベルで構成されるソースコードが多かった．このため，テストデータは機能ラベルを 2～10 個含むソースコードを 10 件ずつ，合計 90 件作成した．表 3 はテストデータの例を示しており，線で囲われたコード機能は推定評価のために伏せた箇所を示している．

表 3 実験データ数の内訳

データ数	ロジック・コーパス 生成と学習	テスト
20, 240	20, 150	90

表 4 テストデータの例

No	機能ラベル列
1	['main'] ['eat']
…	…
90	['main'] ['apply'] ['max'] ['dist'] ['reduce'] ['gcd'] ['pow'] ['mod'] ['count'] ['next', 'token']

4.2 実験方法

複数のコード機能をコンテキストを入力として，推定スコアが高い機能ラベルを 3 件出力し，補完機能として適切であるかを評価する．コード推定の精度の評価は次の 4 段階で行う．評価 1 から評価 4 になるにつれて高評価であることを示している．なお，評価 1 から評価 4 の判定は人が行う．

[評価 1] コード機能として合わない

[評価 2] コード機能としておよそ合う

[評価 3] コード機能として合う

[評価 4] テストデータで伏せた機能ラベルと同じ

4.3 実験結果

図 2 に実験結果を示す．図 2 は推定された機能レベルについて，評価 1～4 の各評価における機能ラベル

数を示している.

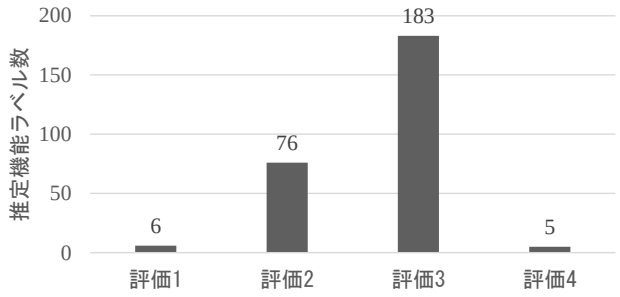

図 2 推定された機能ラベル数

図 2 の結果から，word2vec により学習した NN モデルが補完可能であると推定した機能ラベルは評価 3 のもので 183 個と最も多く，「処理手順として合う機能ラベル」が多く推定されていることが確認できた．一方，評価 4 に相当する「テストデータで伏せた機能ラベル」および評価 1 の「処理手順として合わない機能ラベル」はほとんど推定されなかった．さらに，評価 2 の「処理手順としておよそ合う機能ラベル」が推定される割合は全体の 3 割程度の 76 個であった．

以上，テストデータとして入力した処理手順列に対して，補完可能であると推定された機能ラベルは，中評価から高評価に相当する評価 2 および評価 3 のものがほとんどであった. このことから, 提案手法により，word2vec から得られるコード機能の意味情報に基づいて，処理手順として入力したコード機能に対して，処理手順のコンテキストに適切なコード補完が可能であることが確認できた．

5. まとめ

本研究では，ソースコードにおける処理手順に生じる意味を学習することにより，処理手順間の意味の近さに応じてコード補完を行う手法を提案した．実験では，word2vec から得られるコード機能の意味情報に基づいて，処理手順として入力したコード機能のコンテキストに適切なコード補完が可能であることを示し，提案手法の実現可能性を確認することができた．

今後の課題として， BERT や Transformer などの意味抽出ニューラルネットワークモデルおよび系列変換ニューラルネットワークモデルを適用してコード補完モデルを構築し，性能評価をしていく予定である．

謝辞

本研究は，JSPS 科研費 17K00498 の助成を受けたものである．

参 考 文 献

(1) Li, Y., Choi, D., Chung, J, et al.: “Competition-Level Code Generation with AlphaCode”, arXiv:2203.07814 [cs.PL] (2022)

(2) 齋藤愛莉佳，鷹野孝典: “プログラミング支援のためのコード機能推定に基づくソースコード推薦手法”，人工知能学会 第 123 回知識ベースシステム研究会予稿集，pp.32-pp.37（2021）

(3) 重田智希，鷹野孝典: “プログラミング・ロジックを考慮したソースコード推薦システムの検討”, 第 81 回全国大会講演論文集 2019(1)，pp.305-pp.306（2019）

(4) 内山武尊，新美礼彦: “ソースコード特有の近傍単語の影響を考慮した Word2Vec を用いた類似コード片推薦手法”，ソフトウェアエンジニアリングシンポジウム 2017 論文集，pp.146-pp.153（2017）

(5) Ye, F., Zhou, S., Venkat, A., Marcus, R., Tatbul, N., et al.: MISIM: “A Novel Code Similarity System”, arXiv:2006.05265 [cs.LG] (2020)

(6) Mikolov, T., Chen, K., Corrado, G., Dean, J.: “Efficient Estimation of Word Representations in Vector Space”, arXiv:1301.3781 [cs.CL] (2013)

(7) Alon, U., Zilberstein, M., Levy, O., Yahav, E.: “code2vec:Learning Distributed Representations of Code”, arXiv:1803.09473 (2018)

(8) 山本哲男: “制御構造を考慮したソースコードコーパスに基づくメソッド呼び出し文補完手法”，情報処理学会論文誌 56(2)，pp.682-pp.691（2015）

(9) 山本哲男: “回帰結合ニューラルネットワークを利用した API 推薦手法”, 情報処理学会論文誌 58(4)，pp. 769-779（2017）

(10) Luan, S., Yang, D., Barnaby, C., Sen, K., Chandra, S.: “Aroma: Code Recommendation via Structural Code Search”, arXiv:1812.01158 [cs.SE] (2019)

(11) Kickstarting AI for Code: Introducing IBM's Project CodeNet, https://research.ibm.com/blog/codenet-ai-for-code (2023/01/30 閲覧)

(12) The Java Cookbook, http://javacook.darwinsys.com/, (2023/01/30 閲覧)