Code2Vec for C：C言語を対象としたコードの分散表現の獲得手法の提案

檜枝 琴里[†1,a)]　久住 憲嗣[1,b)]　矢川 博文[†2]　福田 晃[1,c)]

概要：プログラムコードの分散表現を得るための手法として Code2Vec が存在する．これはプログラムコードの埋め込みベクトルを得るために，メソッド本体などのコード片の機能を表すラベルを予測するタスクで学習するものである．これによりプログラムコードにおいてもコード片の意味を考慮した分散表現を得ることが出来る．Code2Vec は Java や C ＃などのオブジェクト指向プログラムを対象としているが，組込みシステム開発ではオブジェクト指向言語ではない C 言語を使用していることが多い．そのため，Code2Vec の手法を適用するために，C 言語からの特徴量の抽出手法が必要であることや，関数名等の命名方法がオブジェクト指向言語と異なるためラベルの予測が困難といった課題がある．そこで本研究では Code2Vec の手法を C 言語プログラムにも対応可能にすべく，C 言語からの特徴量の抽出手法を提案し，関数名等をオブジェクト指向言語と同様のモジュール名と機能名に分解するための ITF-DF 手法を提案する．

1. 序論

自然言語処理では Word2Vec[1] などの意味を考慮した分散表現を得る方法が提案されており様々な応用が展開している．プログラムコードにおいても同様の分散表現を得る手法を使用することにより，ソフトウェア開発を多様な側面から支援することができると考えられるが，手法，応用の両面ともに発展の余地がある．

プログラムコードの分散表現を得る手法として Code2Vec[2] が提案されている．Code2Vec とはプログラムコードの分散表現を得るための手法のひとつであり，コード片を実数ベクトルとして表すことで，コード片やそれ同士の関連性を数値的に表現することができる．Code2Vec により得られた分散表現を用いた主なタスクは，メソッド本体からのメソッド名の推定などがある．この Code2Vec は現状では Java や C#等のオブジェクト指向プログラム言語を対象としている．しかしながら，組込みシステムなどで多く使用されている C 言語はオブジェクト指向言語でないため，Code2Vec を C 言語にそのまま適用できない．特に関数名を推定するタスクにおいては，C 言語の関数名がモジュール固有名と一般的な操作名との両者を含んでいるため，推定したい一般的な操作名のみの推定精度が低下する．

そこで本研究ではこの課題を解決すべく，ITF-DF(Inverse Term Frequency　Document Frequency) を関数名に適用する．ITF-DF 手法とはある文書の中での出現頻度は低いが様々な文書を通して見ると出現頻度が高い単語の重要度が高くなるようにする指標とする．C 言語の関数名に ITF-IDF 適用することにより，関数名を単語に分割した後にそれらをモジュール固有名と一般的な操作名とに選別する．そして，Code2Vec において一般的な操作名のみを学習する．これにより，一つのプログラムコード内で頻繁に出てくる，それ特有でしかないモジュール固有名を削除し，広く一般的に使用される操作名のみを採用することを狙う．重要な操作名のみを関数名として学習することにより，関数名推定の精度向上を図る．

2. 関連研究

プログラムコードの分散表現を得る手法として Code2Vec[2] が提案されている．従来手法では，プログラムコード中に出現する識別子等を学習し，分散表現を得ていた．しかしながらコード中の識別子はほぼ同様でありながらコードの機能が異なることがよくある．例えば，配

[1] 九州大学大学院 システム情報科学研究院
IPSJ, Faculty of Information Science and Electrical Engineering, Kyushu University
[†1] 現在，九州大学大学院 システム情報科学府
Presently with Graduate School of Information Science and Electrical Engineering, Kyushu University
[†2] 現在，富士通九州ネットワークテクノロジーズ (株)
Presently with Fujitsu Kyushu Network Technologies Limited
[a)] hieda@f.ait.kyushu-u.ac.jp
[b)] nel@slrc.kyushu-u.ac.jp
[c)] gakkai.jiro@ipsj.or.jp

表 1　ITF-DF 手法を用いた推定結果

BEFORE	AFTER
ext|get|nojournal	get|nojournal
ext|put|nojournal	put|nojournal
ext|journal|check|start	check|start
ext|journal|start|sb	start|sb
ext|journal|stop	stop
ext|journal|start|reserved	start|reserved
ext|journal|abort|handle	abort|handle
ext|journal|get|write|access	get|write|access
ext|forget	forget
ext|journal|get|create|access	get|create|access
ext|handle|dirty|metadata	handle|dirty|metadata
ext|handle|dirty|super	handle|dirty|super

列に対する操作において，配列中に指定した要素が存在するかどうかを判定するコードと，配列中に指定した要素があった場合にはその値を返すコードとは，ほぼ同じ構造であり返値が違うのみである．そこで，Code2Vec ではこれらの細かな違いを分散表現に反映すべく，コードを抽象構文木にしたのち、木中のふたつの終端記号をつなぐ非終端記号や終端記号の列をパスとして抽出して特徴量とする．この特徴量に基づいて学習することにより，構造がほぼ同じでも細かい差により機能が変わるものを見分けることが可能となった．

3. 解析手法

本節では C 言語のプログラムコードを解析する手順を説明する．まず，Code2Vec を適用するための前処理をする．camel case，_，数値などを区切りとして，関数名を単語に分割し抽出する．これを，用意した全ての C 言語プログラミングコードのファイルを対象に行い，ITF-DF を計算するための辞書とする．ITF-DF の計算方法は log (文書 A における単語 X の出現頻度) / (文書 A における全単語の出現頻度の和) * (全文書数) / (単語 X を含む文書数) とした．この辞書を元に，出現する全ての単語の ITF-DF を計算した後，閾値以下の単語を削除する．残った単語を関数の内容と合わせて DNN で学習する．これにより Code2Vec が適用可能な状態を用意した．

4. 実装

本研究では Python を用いて，関数を libclang で解析し，抽象構文木 (AST) にしている．libclang とは Clang にアクセスするためのライブラリである．AST を探索しパスを抽出することで，これまでの識別子を学習する方法よりも，関数をより高精度で識別することが可能となった．本研究では関数名推定において C 言語を対象としているが，Code2Vec は C 言語のような非オブジェクト指向言語を対象としていないため，コンパイラフロントエンドの Clang とバックエンドの LLVM を用いて C 言語にも Code2Vec を応用できるようにした．これらのコンパイラは，統合開発環境 (IDE) の GUI と連携し開発できるようにしてあり，また C 言語をターゲットとして設計されているため，本研究に用いた．

5. 結果

学習したデータを基に，テスト用に用意した関数の内容をシステムに読み込ませ，その関数名を推定させる．元の関数名との適合率を評価の基準として表す．表 1 は ITF-DF 手法を用いる前後で比較した関数名推定結果である．推定の結果，ITF-DF 手法を用いる以前はモジュール名ごと学習してしまっていたが，ITF-DF 手法を用いることで，モジュール名を除外し，機能名のみを推定することが可能となった．

6. 結論

今回，我々は LLVM と Clang を用いて Code2Vec を C 言語に対応可能な状態にした．またそれを適用する事で C 言語の関数名をモジュール名と機能名に分類し，ITF-DF 手法により重み付けしたものを機械学習することで関数名推定の精度を向上を試みた．結果，ITF-DF 手法を用いることで，汎用的に用いられれいるモジュール名を抜き出すことが可能となり，関数名推定の精度が向上することがわかった．
今後の予定として，より多くのテストデータを用いての詳細な評価を試みる．また現在よりも更に精度が向上する ITF-DF 手法の計算方法を模索する．それを再度評価することで，C 言語特有の拡張が可能となることを期待する．

参考文献

[1] Rong, X.: word2vec parameter learning explained, *arXiv preprint arXiv:1411.2738* (2014).
[2] Alon, U., Zilberstein, M., Levy, O. and Yahav, E.: code2vec: Learning distributed representations of code, *Proceedings of the ACM on Programming Languages*, Vol. 3, No. POPL, p. 40 (2019).