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 TF-IDF 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，TF-IDF(Term Frequency Inverse Document Frequency)[3] を関数名に適用する．TF-IDF 手法は，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値が低くなるため，この値が低いものの重要度を高くする．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)] fukuda@f.ait.kyushu-u.ac.jp

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．

C 言語のプログラムは複数のファイルから構成されているが，ファイル毎に処理を行う．まず，関数の定義を抽出する．関数定義は関数名，引数，返値，関数本体から構成される．

関数名は通常複合語であるため，これから一般的な用語のみを抽出する．単語の区切りを表すために，単語の最初の文字を大文字にそれ以外を小文字にするキャメルケースや，_が用いられるため，それらの区切りで単語を抽出する．さらに，数値は一般語のとしては不適切であることが多いため，数値を削除する．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，すべての単語についての出現文書数 (DF) を計算し，全体の辞書とする．また，各ファイルごとに単語毎の出現頻度 (TF) の辞書を作成する．これらの辞書に基づいて TF-IDF を計算する．文書 A における単語 X の TF-IDF は，(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) * log(全文書数) / (単語 X を含む文書数) で計算する．出現する全ての単語の TF-IDF を計算した後，閾値以下の単語を削除した関数名を作成する．

次に関数本体から特徴量を抽出する．Code2Vec の他言語用の実装に習って解析する．関数本体を構文解析して抽象構文木に変換し，木中の終端記号すべてを抽出する．抽出した終端記号のすべての組み合わせを作成する．組となる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する．これを特徴量とする．

この一般語のみで構成した関数名と，関数本体から抽出した特徴量とを Code2Vec を用いて学習する．

4. 実装

本研究では前節で説明した過程を Python を用いて実装する．また，C 言語を構文解析して抽象構文木にするために，clang[4] を用いる．表 1 に使用環境を示す．

表 1 開発環境

開発環境	バージョン
MacOS Sierra	10.12.6
Python	3.7.0
Clang	900.0.39.2
LLVM	9.0.0

表 2 TF-IDF 手法を用いた推定結果

TF-IDF使用前	TF-IDF使用後
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

5. 結果

関数名から一般名を抽出できているかを確認するために，実際のプログラムを対象に TF-IDF を計算し，閾値以下の単語を削除して関数名を再構成する．対象としたプログラムは linux 4.20 であり，逆文書頻度 (IDF) はカーネル全体を対象として計算する．また、単語の出現頻度 (TF) は fs/ext4/ext4_jbd2.c のみを対象として計算する．また，閾値は 1.0 とした．その結果を，表 2 に示す．処理前の関数名は単語に分割したものをで結合した結果である．また，処理後は TF-IDF による一般語抽出をした結果である．TF-IDF による一般語抽出を実施する前はモジュール名が関数名に含まれるが，TF-IDF 手法を用いることで，モジュール名を除外し，機能名のみを抽出することができた．

6. 結論

本研究では Code2Vec を C 言語に適用すべく，LLVM と Clang を用いて特徴量を抽出する実装について示した．また，関数本体から関数名を推定するタスクにおいては，C 言語の関数名などの識別子は，モジュール名と一般的な機能名との複合語で構成されることが多いため，学習の際に障害になる可能性があることを議論した．さらに，この問題を解決すべく，C 言語の関数名を TF-IDF を用いてモ

ジュール名と機能名に分類する手法を提案した．その結果，C 言語においても関数名から一般的な機能名のみを抽出することができることを示した．

今後の課題としては，提案手法を適用することにる評価，関数名のみではなく変数名等の多の識別子への適用による評価などが挙げられる．

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).
[3] Ramos, J. et al.: Using tf-idf to determine word relevance in document queries, *Proceedings of the first instructional conference on machine learning*, Vol. 242, Piscataway, NJ, pp. 133–142 (2003).
[4] Lattner, C. et al.: clang: a C language family frontend for LLVM, *Website* (2007).