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?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? 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であり，コード片を高次元の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．これによりコードを抽象構文木にして扱うことができるため，コードの終端記号同士間のパスを特徴量として，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．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．具体的には，関数名がモジュール固有名も操作名も含んでいるため，モジュール固有名を重要視してしまうことや，マクロ使用を関数名と混合してしまうなどの課題がある．

そこで本研究ではこの課題を解決すべく，TF-IDF 手法 (…) を関数名に適用する．TF-IDF 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ものである．これによりモジュール固有名と操作名とを選別し，操作名のみを学習する手法を提案する．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の重要度を下げ，広く一般的に使用される操作名の重要度を上げることを狙う．重要な操作名に重みがついた状態で学習することにより，関数名推定の精度向上を図る．

具体的には，C 言語で記述されたプログラムコードをいくつか用意し，ITF-DF 手法を用いて重み付けをした関数名を学習する．学習したコード片に基づき C 言語にも対応させた Code2Vec を用いることで，コード片をベクトルとし，関連のあるコード片同士を紐付ける．評価の方法として，評価をするためのプログラムコードを用意し，その中の関数名と，関数名推定を行った結果とを照らし合わせ，元の関数名との適合率を表す．

[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

Original name: ioeventfd_release	Original name: ioeventfd_in_range	Original name: ioeventfd_write
(0.231438) predicted: ['ioctl', 'from', 'context']	(0.390824) predicted: ['i', 'control']	(0.263276) predicted: ['pci', 'set', 'size']
(0.204751) predicted: ['dm', 'cache', 'get', 'size']	(0.195847) predicted: ['arch', 'notify']	(0.130477) predicted: ['verify', 'rate']
(0.181396) predicted: ['snd', 'init', 'port']	(0.086416) predicted: ['i', 'control', 'clock']	(0.106973) predicted: ['debug', 'off']
(0.084475) predicted: ['return', 'entry']	(0.065557) predicted: ['g', 'i', 'c', 'pad', 'mode']	(0.085039) predicted: ['clk', 'rate', 'div', 'range', 'iter']
(0.063291) predicted: ['port', 'read', 'status']	(0.055883) predicted: ['i', 'rate', 'cb']	(0.079718) predicted: ['is', 'hw']
(0.053508) predicted: ['set', 'hw', 'mode', 'port']	(0.049539) predicted: ['add', 'one', 'stat']	(0.073356) predicted: ['pm', 'enable', 'port']
(0.048751) predicted: ['s', 'be']	(0.044493) predicted: ['max', 'pcm', 'rx']	(0.073022) predicted: ['do', 'single']
(0.048549) predicted: ['sync', 'idx']	(0.044365) predicted: ['v', 'regulator', 'set', 'mode']	(0.063968) predicted: ['get', 'hw', 'info']
(0.042842) predicted: ['destroy', 'setup']	(0.036390) predicted: ['a', 'hw', 'params']	(0.063215) predicted: ['queue', 'stack', 'map', 'free']
(0.041000) predicted: ['port', 'read', 'data']	(0.030685) predicted: ['acpi', 'valid', 'internal', 'object']	(0.060955) predicted: ['class', 'get']
Attention:	Attention:	Attention:

2. 解析手法

本研究では C 言語で書かれているプログラムコードをいくつか用意し，その中の関数名を，Code2Vec を用いて分散表現し，モジュール名と機能名に分類した．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．LLVM は，中間言語を介して対象のアーキテクチャに最適なマシン語へ変換することができるため，多様なプラットフォームで最適なマシン語へ変換することができる．Clang は LLVM 上で動作することを意図し設計されている．これらのコンパイラは，統合開発環境 (IDE) の GUI と密接に連携し開発できるようにしてあり，様々な環境でのクロスプラットフォーム開発を容易にし，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．
モジュール名と機能名に分類した関数名を，ITF-DF 手法により 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) と計算し，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重み付けをした．さらにモジュール名と機能名に分類し重み付けをした関数名を，関数の内容と紐づけて学習した．これを辞書として用意し，関数の内容からその関数名として適切だと思われるものを，辞書から選び提案する．

3. 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 は関数名推定結果の一部である．推定の結果，テストデータとして用意していた C 言語プログラムコードにあらかじめ設定されていた関数名と同じ関数名を推定することができた．しかしその精度は必ずしも高いとは言えず，全く別の関数名を推定する場合もあった．

4. 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．

5. 参考文献