様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２７ 年　 ６ 月 １９ 日現在

機関番号：１４３０３
研究種目：基盤研究(C)
研究期間：2012～2014
課題番号：２４５０００３８

研究課題名（和文）クラウドを利用したソースコード中の不具合検出機構の提案

研究課題名（英文）Development of fault-prone prediction model using web cloud system

研究代表者
水野　修（Mizuno, Osamu）

京都工芸繊維大学・工芸科学研究科・准教授

研究者番号：６０３１４４０７

交付決定額（研究期間全体）：（直接経費）　　4,100,000円

研究成果の概要（和文）：本研究ではインターネット上から自動的にデータを収集し，ソフトウェア中の不具合混入モジュールを自動的に発見するシステムの開発を目指した．そのため，ソースコード中の識別子の語彙を「トピック」として扱い，不具合に関連の深いトピックを自動抽出して不具合情報と結びつける手法の提案を実施した．また，そうした場合に省略された識別子が大量に発見されるため，省略識別子を標準形に復元する手法を深層学習を用いることで実現した．

研究成果の概要（英文）：In this research, we proposed a system to predict fault-prone modules in software source code. To do so, we focused on the vocabulary of the identifiers in source code. First, we developed a system to find topics in the faulty source code. Second, we developed a system to restore abbreviated identifiers to the original identifiers using a deep learning technique. By these models, we found the ways to utilise identifiers in software for fault-prone module prediction.

研究分野：ソフトウェア工学

キーワード：ソフトウェア　ソースコード　リポジトリマイニング　識別子　トピック分析

１．　研究開始当初の背景

高品質なソースコードの作成はプロダクトの品質向上だけでなくコストの削減にもつながる．コードを作成した時点で Fault-prone モジュールを特定できれば，早期にバグを除去できるだけでなく，レビュー，デバッグに費やす工数の削減も可能となる．そのため，これまでにも Fault-prone モジュールを予測すべく，多くの研究が行われてきた[1]．従来の手法では，主にモジュールの複雑さや変更頻度などのソフトウェアメトリクスを用いて予測モデルを構築している．しかし，こうしたソフトウェアメトリクスを測定するためには，メトリクスの測定環境が必要となる．また，構築された予測モデルが別の環境で利用できるという保証も無かった．

これに対して，我々はソースコードのみを入力として与え，メトリクスなどの測定無しに Fault-prone モジュールの予測が可能となる手法を提案している[2]．この手法では，迷惑メール検出に利用されるスパムフィルタで利用される技術をソフトウェアのソースコードに対して適用し，純粋にコードのテキスト情報のみから Fault-prone モジュールを予測する．この手法を「Fault-prone フィルタリング」と呼ぶ．我々は，スパムフィルタの考え方がソースコード内の不具合についても適用できるのではないかと考えた．すなわち，ソースコードをテキストの語彙に分割し入力することで，ベイズ識別器によりその中に不具合が存在するか否かを判定できるのではないかと考えた．また，同時にソフトウェア開発の「履歴」に着目し，履歴を含むリポジトリからのデータマイニングを通じて不具合を引き起こす原因を探ってきた．

これまでの研究ではこの手法の理論的な整備と小規模な実験を通じて有効性の確認を行ってきている．

２．研究の目的

Fault-prone フィルタリングの研究を進めるうちに，ソースコードをテキスト情報として考えるアプローチでは，入力しているテキストの語彙情報に 2 つの側面があることに思い当たった．それは，(a) 局所的に不具合を引き起こす原因となる語彙群と，(b) 大局的に不具合を引き起こす原因となる語彙群である．このうち，(a)に関しては単一のプロジェクトからの履歴を探索することで収集，利用できることがこれまでの研究で明らかになっている．一方，(b)に関しては多くのソフトウェアプロジェクトからのデータを収集する必要があるため，未だその効果を解明できていない．そこで，本研究では(b)の大局的に不具合を引き起こす原因となる語彙群を明らかにすべく，大規模な語彙情報を収集する実験設備を整えた上で研究を実施した．

また，インターネット上からデータを自動的に収集しながら，不具合予測結果を返す機構を開発する．これによって，世界中で開発されている多数のオープンソースソフトウェアのリポジトリから大規模に語彙情報を収集することを目指す．

３．研究の方法

不具合の原因となる語彙群を推定するために 2 つのアプローチを実施した．

(1) トピック分析によるバグ混入コミットの分析

一般的に，開発者はプログラム理解のためにソースコードの識別子にプログラムの内容や機能に関するキーワードを埋め込むため，ソースコードから識別子を取り出しそれらにトピック分析を適用すれば，プログラムに潜在するトピックが推定可能である．本研究では，これの対象をバグ混入コミットで変更されたソースコードに限定する．

(2) 深層学習による省略識別子の原義推定

ソースコードの可読性はソフトウェアを開発・保守する上で重要な要素である．識別子はソースコードを正しく理解するための重要な情報源であるが，コーディング時にプログラマの判断でしばしば省略される場合がある．識別子の過度な省略は初見の読者にとって理解の妨げとなりうる．本報告では省略された識別子の元となった識別子を推定する手法を提案する．

近年，深層学習を用いて自然言語を学習し，単語の意味を高次元のベクトルで表せるツール Word2Vec が注目されている．提案手法では Word2Vec を用いてソースコードから識別子の意味を表すベクトルに変換し，ベクトル間の距離を比較する事で元の識別子を推定する．

４．研究成果

(1) 本実験で対象としたのは Apache プロジェクトで開発されているソフトウェア「Apache MINA」の開発データと，O/R マッピング・フレームワーク「Cayenne」の二つである．Apache MINA はネットワークアプリケーションの開発フレームワークであり，ユーザが簡単に高度なネットワークアプリケーションを作成できるようにするための補助ツールである．Cayenne はコミュニティ「ObjectStyle」から無償で提供されているオープンソース・ソフトウェアである．

具体的には，ソースコードの git リポジトリとバグデータベース JIRA から抽出したバグが混入した時点などの情報を用いる．

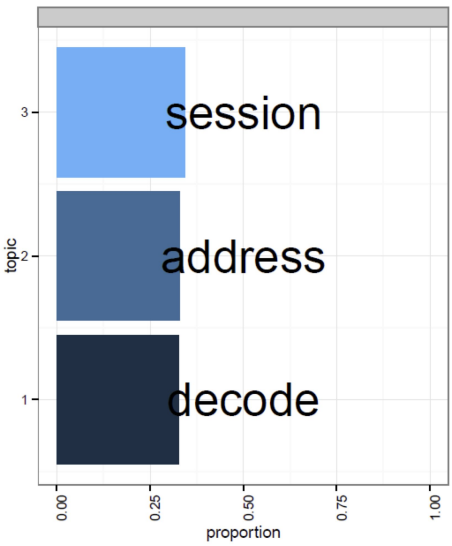

図 1: MINA プロジェクトの全体トピック

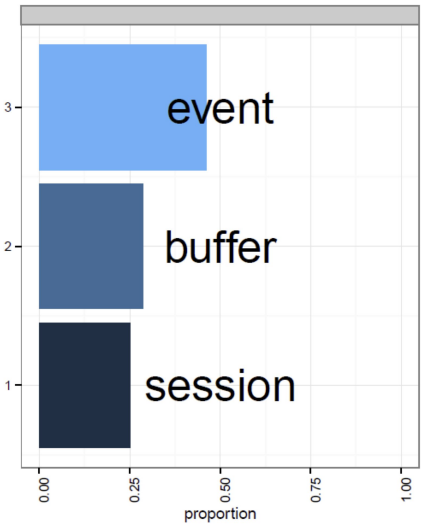

図 2: MINA プロジェクトのバグ混入コミットからのトピック

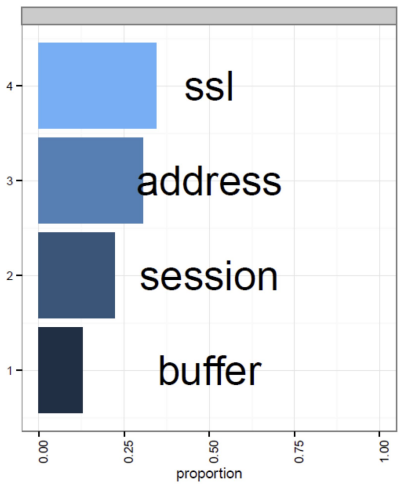

図 3: MINA プロジェクトのバグ非混入コミットからのトピック

図 1，2，3 は Apache MINA プロジェクトの全体，バグ混入，バグ非混入それぞれのコミットから抽出したトピックの最も重要な語を表したものである．図 1 にプロジェクト全体から抽出したトピックを挙げる．ここで挙げられた ``session''，``address''，``decode''の 3 語が，MINA では支配的であることを示している．以降のトピックを示すグラフの横軸は正規化したトピックの出現頻度を示しており，縦軸はトピックの種類を表す．また，グラフに重ねてトピックに含まれる単語を示している．図 1 中のトピックに含まれる単語はネットワークアプリケーションの開発フレームワークという特性を反映し，session や address といった語が多く出現する．
図 2 にプロジェクト MINA におけるバグコミットから抽出した頻出トピックを挙げる．図 2 から は “event”，”buffer”，“session”というトピックが出現し，中でも “event”に関するバグ混入コミットが多いことが分かる．一方，図 3 にプロジェクト MINA におけるバグ非混入コミットから抽出したトピックを示す．このグラフでは，“ssl”が最頻出であり，“address”，“session”，“buffer”と続く．
このことから，プロジェクト MINA におけるバグが混入するコミットでは “event”に関するトピックが多く，逆に “ssl”に関連するトピックが存在したコミットではバグが混入しないことが多いと分かる．

(2) 本研究では Apache ANT，Apache MINA および EC-CUBE の 3 個のオープンソースプロジェクトを利用する．
これらのプロジェクトより，識別子のみを抽出し，3 文字以下のものを省略識別子と考えて，これらの意味を推定する実験を行った．推定標準識別子が正解であるかの判定は本研究室のプログラミング経験のある大学院生が手作業で行った．復元候補の第 16 位まで正解であるかの判定をしたが，チェックの手間を省くために識別子間の意味ベクトルの距離が 0.2 未満の候補を機械的に省いている．

図 4 に MINA プロジェクトにおける代表的な省略識別子と，その推定された標準形を示す．提案手法では multi-word の復元の精度が優れている．3 文字で 3 文字目が「e」の省略識別子の上位の標準識別子の候補に文字列「exception」が含まれているものが複数存在した．
これは word2vec による学習において「exception」の意味を十分に学習できていることを示している．また，識別子は省略されてもソースコードの文脈において同じ役割を果たしていると言える．

分類	省略形	推定標準形	順位	距離
single	obj	putobject	1	0.315
		expiringobject	2	0.298
		expobject	3	0.262
		testinheritedobjectserialization	4	0.234
	pos	position	1	0.421
		responsepos	3	0.349
		getoverflowposition	4	0.346
	len	destlength	1	0.399
		byte_digest_length	2	0.368
		byte_block_length	3	0.351
		encodeinitialgreetingpacket	4	0.339
	ctx	getcontext	1	0.529
		context	2	0.482
		gss_context	3	0.450
		createcontext	4	0.434
multi	baf	bytearrayfactory	1	0.723
		getbytearrayfactory	2	0.612
		simplebytearrayfactory	3	0.600
	tmf	trustmanagerfactorykeystore	1	0.845
		trustmanagerfactoryalgorithm	2	0.826
		trustmanagerfactoryparameters	3	0.812
		trustmanagerfactoryprovider	4	0.807
	nfe	numberformatexception	1	0.813
	pde	protocoldecoderexception	1	0.614
	pee	protocolencoderexception	1	0.672

図 4: MINA プロジェクトにおける省略識別子の代表的な復元

まとめ

本研究ではインターネット上から自動的にデータを収集し，ソフトウェア中の不具合混入モジュールを自動的に発見するシステムの開発を目指した．そのため，ソースコード中の識別子の語彙を「トピック」として扱い，不具合に関連の深いトピックを自動抽出して不具合情報と結びつける手法の提案を実施した．また，そうした場合に省略された識別子が大量に発見されるため，省略識別子を標準形に復元する手法を深層学習を用いることで実現した．

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 4 件）

[1] Osamu Mizuno and Hideaki Hata, "A Metric to Detect Fault-Prone Software Modules Using Text Classifier," International Journal of Reliability and Safety, 7(1), pp. 17-31, February 2013. 査読有.
[2] 畑 秀明, 水野 修, 菊野 亨, "開発履歴メトリクスを用いた細粒度な Fault-prone モジュール予測," 情報処理学会論文誌, 53(6), pp. 1635-1643, 2012 年 6 月. 査読有.
[3] Osamu Mizuno and Michi Nakai, "Can Faulty Modules Be Predicted by Warning Messages of Static Code Analyzer?," Advances in Software Engineering, 2012(924923), 8 pages, May 2012. 査読有.
[4] 畑 秀明, 水野 修, 菊野 亨, "不具合予測に関するメトリクスについての研究論文の系統的レビュー," コンピュータソフトウェア, 29(1), pp. 106-117, 2012 年 2 月. 査読有.

〔学会発表〕（計 14 件）

[1] 岡嶋 秀記, 水野 修, "機械学習を用いた省略識別子の復元手法," 電子情報通信学会技術研究報告, 114(SS2014-68), pp. 79-84, 2015 年 3 月 9 日. (沖縄県青年会館, 那覇市, 沖縄県)
[2] 岡嶋 秀記, 河端 駿也, 水野 修, "単語ベクトルを用いた省略識別子の復元手法," ソフトウェア信頼性研究会 FORCE2014 予稿集, 2-2, 2014 年 12 月 6 日. （尾道商工会議所記念館, 尾道市, 広島県）
[3] Osamu Mizuno and Yukinao Hirata, "A Cross-Project Evaluation of Text-Based Fault-Prone Module Prediction," In Proc. of 6th International Workshop on Empirical Software Engineering in Practice (IWESEP2014), pp. 43-48, November 12, 2014. (Osaka University, Suita-shi, Osaka, Japan) (Acceptance rate: 56%, 10/18)
[4] Yukiya Uneno and Osamu Mizuno, "Identifying Bug Injected Files from Bug Description Using Word2vec," In Poster presentation of 6th International Workshop on Empirical Software Engineering in Practice (IWESEP2014), November 12, 2014. (Osaka University, Suita-shi, Osaka, Japan)
[5] Hideki Okajima and Osamu Mizuno, "Applying Vector Calculation for Identifiers in Source Code Towards Bug Prediction," In Poster presentation of 6th International Workshop on Empirical Software Engineering in Practice (IWESEP2014), November 12, 2014. (Osaka University, Suita-shi, Osaka, Japan)
[6] Akihisa Yamada and Osamu Mizuno, "A Text Filtering Based Approach to Classify Bug Injected and Fixed Changes," In Proc. of 12th International Conference on Software Engineering Research, Management and Applications (SERA2014), pp. 680-686, August 31, 2014. (Kitakyushu International Conference Center, Kitakyushu-shi, Fukuoka, Japan) (Acceptance rate: 59%, 19/32)
[7] Naoki Kawashima and Osamu Mizuno, "Predicting Fault-Prone Modules by Word

Occurrence in Identifiers," In Proc. of 12th International Conference on Software Engineering Research, Management and Applications (SERA2014), Studies in Computational Intelligence , 578, pp. 87-98, August 31, 2014. (Kitakyushu International Conference Center, Kitakyushu-shi, Fukuoka, Japan) (Acceptance rate: 59%, 19/32)
[8] Osamu Mizuno and Junwei Liang, "Does a Code Review Tool Evolve as the Developer Intended?," In Proc. of 12th International Conference on Software Engineering Research, Management and Applications (SERA2014), Studies in Computational Intelligence, 578, pp. 59-74, August 31, 2014. (Kitakyushu International Conference Center, Kitakyushu-shi, Fukuoka, Japan) (Acceptance rate: 59%, 19/32)
[9] 山田 晃久, 水野 修, "バグを混入・除去するソースコード差分の判定手法の提案," ソフトウェアシンポジウム 2014 論文集, 07_研究論文, pp. 56-64, 2014 年 6 月 8 日. (秋田市にぎわい交流館 AU, 秋田市, 秋田県)
[10] 川島 尚己, 水野 修, "識別子中の単語情報を用いた Fault-prone モジュール予測," ソフトウェアシンポジウム 2014 論文集, pp. 72-80, 2014 年 6 月 8 日. (秋田市にぎわい交流館 AU, 秋田市, 秋田県)
[11] 椋代 凜, 水野 修, "オープンソースソフトウェアにおけるバグ混入コミットのトピック分析," ソフトウェア信頼性研究会第 9 回ワークショップ(FORCE2013)論文集, 2013 年 12 月 11 日. (愛媛大学, 松山市, 愛媛県)
[12] Osamu Mizuno, "On Effects of Tokens in Source Code to Accuracy of Fault-Prone Module Prediction," In Proc. of the 17th International Computer Science and Engineering Conference (ICSEC2013), 103 - 108, September 4, 2013. (Bangkok, Thailand) (Acceptance rate: 57%, 73/128)
[13] Kimiaki Kawamoto and Osamu Mizuno, "Predicting Fault-Prone Modules Using the Length of Identifiers," In Proc. of 4th International Workshop on Empirical Software Engineering in Practice (IWESEP 2012), pp. 30-34, October 26, 2012. (Osaka University, Osaka-shi, Osaka, Japan) (Acceptance rate: 8/14, 57%)
[14] Hideaki Hata, Osamu Mizuno, and Tohru Kikuno, "Bug Prediction Based on Fine-Grained Module Histories," In Proc. of 34th International Conference on Software Engineering (ICSE2012), pp. 200-210, June 2, 2012. (Zurich, Switzerland) (Acceptance rate: 21.3%, 87/408)

６．研究組織

(1)研究代表者

水野 修 （Osamu Mizuno）
京都工芸繊維大学・大学院工芸科学研究科・准教授
研究者番号：60314407