

CPC: Automatically Classifying and Propagating Natural Language Comments via Program Analysis

Juan Zhai
juan.zhai@rutgers.edu
Purdue University, Rutgers University

Guanhong Tao
taog@purdue.edu
Purdue University

Lei Xu[†]
xlei@nju.edu.cn
Nanjing University

Xiangzhe Xu
xxz@smail.nju.edu.cn
Nanjing University

Minxue Pan[†]
mxp@nju.edu.cn
Nanjing University

Weifeng Zhang
zhangwf@njupt.edu.cn
Nanjing University of Posts and
Telecommunications

Xiangyu Zhang
xyzhang@cs.purdue.edu
Purdue University

Yu Shi
shi442@purdue.edu
Purdue University

Shiqing Ma
shiqing.ma@rutgers.edu
Rutgers University

Lin Tan
lintan@purdue.edu
Purdue University

ICSE '20: Proceedings of the ACM/IEEE
42nd International Conference on Software Engineering
Pages 1359–1371

背景

- 詳細なコメントを適切に配置することで保守性, 信頼性の向上につながる
- 既存のコメントやコードの情報を用いて, 適当にコメントを伝播する手法は確立されていない

目的

- 包括的なコメントの分類手法の構築
- 分類結果を用いたコメント伝播手法の構築

分類手法の構築

- コメントのサンプリング
 - コードエンティティで層化抽出
- コメントへのタグ付け
 - サンプリングされたコメントがどの観点で書かれたものかをタグ付けする
 - 4人の参加者

コードエンティティと観点

- コードエンティティ
 - クラス, メソッド, 文, 式
- 観点
 - What, Why, How-it-is-done, Property, How-to-use

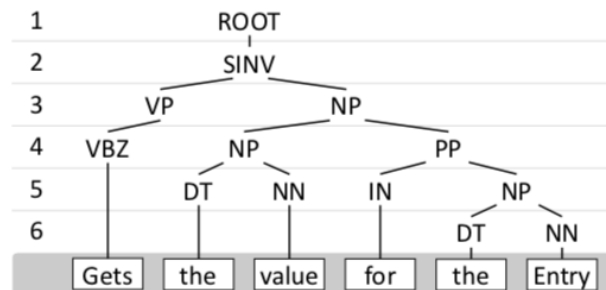
分類法の実装

- 前処理
 - 分散処理（word2vec）の実行
 - 固有名詞の置換とストップワードの除去
- 特徴量の抽出
- 分類アルゴリズムの適用
 - 決定木，ランダムフォレスト，CNNの3つのアルゴリズムで試す

データセット

- プロジェクト
 - JDK 8
 - Guava
 - Apache Commons Collections
 - Joda
- 規模
 - クラス数：450 - 2500
 - 行数：43 - 310KLOC
 - 行数の30%がコメント

特徴量の詳細



(a) Stanford Parse Tree

1: ROOT; 2: SINV; 3: VP NP; 4: VBZ NP PP; 5: DT NN IN NP; 6: DT NN

(b) The BFS Sequence of Stanford Parse Tree

Figure 4: Parse Tree and BFS

Table 2: Features for Comments Classification

Feature	Type	Description
tokenNum	numeric	number of tokens in a cleaned comment
classNum	numeric	number of classes mentioned in a cleaned comment
tree	string	a depth first traversal of Stanford parse tree
NPNum	numeric	number of noun phrases in parse tree
VPNum	numeric	number of verb phrases in parse tree
PPNum	numeric	number of prepositional phrases in parse tree
caseNum	numeric	number of case marking relations
preconjNum	numeric	number of preconjunction relations

分類精度

- 5回交差検証を実行

Table 3: Comment Classification Result

	Perspective			Code Entity		
	DTC	RFC	CNN	DTC	RFC	CNN
Precision	87.84%	87.78%	95.15%	97.39%	98.09%	89.33%
Recall	95.22%	91.39%	93.78%	99.27%	99.27%	75.28%
F1 Score	93.43%	93.17% _s	94.46%	98.55%	98.90% _s	81.71%
Hamming Loss	0.014583	0.014583	0.011979	0.010417	0.0007813	0.0674157

コメント伝播

- コードエンティティ, 観点毎に伝播ルールを作成

Class-level Propagation Rules:		Class-level Propagation Rules:			
PROPERTY-INSTAN	$\frac{propComment(o, c) \quad instantiation(v, o)}{propComment(v, c[v/o])}$	WHAT-INHER	$\frac{whatComment(q, c) \quad inheritance(p, q)}{whatComment(p, c[p/q])}$		
PROPERTY-INHER	$\frac{propComment(q, c) \quad inheritance(p, q)}{propComment(p, c[p/q])}$	WHAT-IMPL	$\frac{whatComment(i, c) \quad implementation(o, i)}{whatComment(o, c[o/i])}$		
PROPERTY-IMPL	$\frac{propComment(i, c) \quad implementation(o, i)}{propComment(o, c[o/i])}$	Method-level Propagation Rules:			
Method-level Propagation Rules:		WHAT-CALLEE2ST	$\frac{whatComment(m, c) \quad invocation(s, m)}{whatComment(s, c[ap/fp])}$		
PROPERTY-CALLEE2ST	$\frac{propComment(m, c) \quad invocation(s, m)}{propComment(s, c[ap/fp])}$	WHAT-METHOD2ST	$\frac{whatComment(m, c) \quad lastSt(m, s) \ \&\&(preSts == \emptyset \ \ preSts \subseteq exSts)}{whatComment(s, c)}$		
PROPERTY-METHOD2ST	$\frac{propComment(m, c) \quad onlySt(s, m)}{propComment(s, c)}$	Statement-level Propagation Rules:			
Statement-level Propagation Rules:		WHAT-ST2CALLEE	$\frac{whatComment(s, c) \quad invocation(s, m) \ \&\& noOtherOp(s)}{whatComment(m, c[fp/ap])}$		
PROPERTY-ST2CALLEE	$\frac{propComment(s, c) \quad invocation(s, m) \ \&\& noOtherOp(s)}{propComment(m, c[fp/ap])}$	WHAT-ST2METHOD	$\frac{whatComment(s, c) \quad lastSt(m, s) \ \&\&(preSts == \emptyset \ \ preSts \subseteq exSts)}{whatComment(m, c)}$		
PROPERTY-ST2METHOD	$\frac{propComment(s, c) \quad contain(m, s) \ \&\& ap \subseteq fp}{propComment(m, c[fp/ap])}$	Variable-level Propagation Rules:			
Variable-level Propagation Rules:		WHAT-DEFUSE	$\frac{whatComment(v, c) \quad defUse(u, v)}{whatComment(u, c)}$		
PROPERTY-DEFUSE	$\frac{propComment(v, c) \quad defUse(u, v)}{propComment(u, c)}$	(b) What Propagation Rules			
PROPERTY-CONTAINER	$\frac{propComment(l, c) \quad contain(e, l)}{propComment(e, c[e/l])}$				
(a) Property Propagation Rules					
Method-level Propagation Rules:					
HOW-CLONE	$\frac{howComment(m, c) \quad clone(m', m, 100\%)}{howComment(m', c)}$	Note:			
HOW-DIFFTYPE	$\frac{howComment(m, c) \quad clone(m', m, 90\%) \ \&\& diffType(m', m)}{howComment(m', c[t'/t])}$	<i>c</i>	comment	<i>u</i>	code that uses a variable <i>v</i>
		<i>o</i>	class	<i>ap</i>	actual parameters
		<i>p</i>	subclass	<i>fp</i>	formal parameters
		<i>q</i>	superclass	<i>l</i>	container variable
		<i>i</i>	interface	<i>e</i>	element contained in the container <i>l</i>
		<i>m/m'</i>	method	<i>t/t'</i>	type
		<i>s</i>	statement	<i>preSts</i>	statements before the current statement
<i>v</i>	variable	<i>exSts</i>	exception-handling statements		
(c) How-it-is-done Propagation Rules					

Figure 5: Comment Propagation Rules

コメント伝播

- Property-Callee2St

- $$\frac{\text{propComment}(m,c) \text{ invocation}(s,m)}{\text{propComment}(s,c[ap/fp])}$$

Note:		
<i>c</i>	comment	<i>u</i> code that uses a variable <i>v</i>
<i>o</i>	class	<i>ap</i> actual parameters
<i>p</i>	subclass	<i>fp</i> formal parameters
<i>q</i>	superclass	<i>l</i> container variable
<i>i</i>	interface	<i>e</i> element contained in the container <i>l</i>
<i>m/m'</i>	method	<i>t/t'</i> type
<i>s</i>	statement	<i>preSts</i> statements before the current statement
<i>v</i>	variable	<i>exSts</i> exception-handling statements

- コメント *c* が呼び出し先メソッド *m* に関する特性が含まれ、*m* が文 *s* で呼び出される場合、コメント *c* は呼び出し先メソッド *m* から文 *s* に伝播する

コメント伝播手法(CPC)の評価

1. 観点ごとに効果的にコメントを伝播できているか
2. CPCは開発者にどれほど役立つか
3. CPCはコメントを改良しているか
4. CPCはバグを検出するのに効果的か

観点ごとに効果的にコメントを伝播できているか

Table 5: Comment Propagation Accuracy

Perspective	Accuracy		
	dist=0	dist<0.5	dist $\geq$ 0.5
Property	100.00%	76.00%	85.00%
What	100.00%	71.00%	70.00%
How-it-is-done	100.00%	75.00%	70.00%

Table 4: Comment Propagation Summary

Perspective	Project	Similarity with Existing Comments									
						dist=0		dist<0.5		dist $\geq$ 0.5	
		#c	#m	#ec	#pc	#cmt	%	#cmt	%	#cmt	%
Property	JDK	998	17727	21147	39274	9133	75.11%	2191	18.02%	835	6.87%
	Collections	247	2687	3151	4222	1301	73.30%	372	20.96%	102	5.75%
	Guava	518	6140	1940	8425	2718	88.28%	259	8.41%	102	3.31%
	Joda-Time	219	5011	2344	4393	1313	80.50%	111	6.81%	207	12.69%
	ApacheDB	193	3508	1898	2552	779	82.43%	57	6.03%	109	11.53%
What	JDK	628	10841	12927	5029	1368	39.66%	1550	44.94%	531	15.40%
	Collections	70	989	1472	330	105	44.30%	83	35.02%	49	20.68%
	Guava	205	2847	1347	1294	419	49.47%	333	39.31%	95	11.22%
	Joda-Time	83	1725	1949	885	237	29.40%	325	40.32%	244	30.27%
	ApacheDB	78	1426	1316	682	169	29.14%	366	63.10%	45	7.76%
How-it-is-done	JDK	261	974	1392	16285	15516	96.72%	394	2.46%	133	0.83%
	Collections	41	98	100	113	53	67.09%	22	27.85%	4	5.06%
	Guava	20	33	31	127	108	85.71%	16	12.70%	2	1.59%
	Joda-Time	15	22	29	130	32	35.20%	37	29.13%	58	45.67%
	ApacheDB	180	285	254	519	421	84.04%	58	7.39%	22	4.39%

CPCは開発者にどれほど役立つか

- 14人に伝播元コメント，伝播コメント（80件づつ）を実際に読んでもらう
- 偏りを避けるために伝播元コメントと伝播コメントを混ぜて読ませるようにする
- 理解度，コードとの内容一致度，コメントの内容の自然さを1~5で評価してもらう

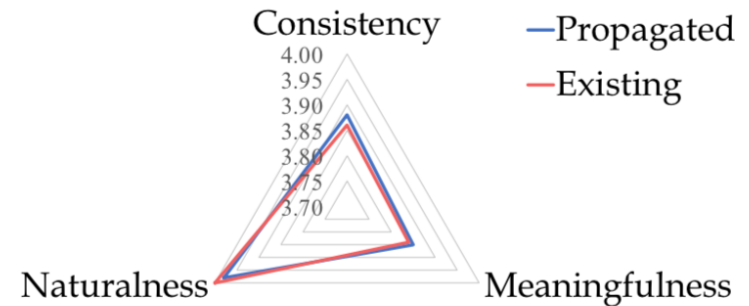


Figure 6: User Study Comparison Result

CPCはコメントを改良しているか

- 新たに生成されたコメント, 不完全なコメント, 欠陥を引き起こす可能性のあるコメントを評価

Table 6: Comment Propagation Improvement

Perspective Project	Property			What			How-it-is-done		
	#N	#I	#W	#N	#I	#W	#N	#I	#W
JDK	26862	11	243	1580	1	0	242	n.a.	0
Collections	2404	11	42	93	0	0	34	n.a.	0
Guava	5344	0	2	447	0	0	1	n.a.	0
Joda-Time	2757	0	5	79	0	0	3	n.a.	0
ApacheDB	1607	0	0	102	0	0	18	n.a.	0

CPCはバグを検出するのに効果的か

- 伝播されたコメントを持つコードから以下の例外を発生させるバグに絞って件数を求める
 - NullPointerExceptionとIndexOutOfBoundsException
- 57件のうち実際にバグと思われるものが37件, そのうち30件のバグを報告して修正が加えられた

Table 7: Code Bug Detection

Project	Version	#Bugs	Buggy Method	Confirmed
Collections	4.2	29	CompositeCollection.iterator()	Yes
			CompositeMap.removeComposited(final Map<K, V>)	Yes
			...	
Guava	28.0	6	Throwables.getRootCause(Throwable)	No
			...	
ApacheDB	3.2	2	Utilities.printClasspath()	Yes
			ConsoleFileOutput.getDirectory()	No

まとめ

- コメントの包括的な分類手法を構築
- 分類結果をもとにコメント伝播手法を構築
- コメント伝播によってバグも検出できることがわかった

所感

- コメントの重要性を改めて感じた
- バグ検出の支援にもなるのはかなり有用であると思った
- 既存のコメント自動生成と性能の差が気になる