

Towards Automated Classification of Code Review Feedback to Support Analytics

Asif Kamal Turzo ♡, Fahim Faysal ♣, Ovi Poddar ♣ Jaydeb Sarker ♡

Anindya Iqbal ♣ Amiangshu Bosu ♡

♡ *Wayne State University, Detroit, Michigan, USA*

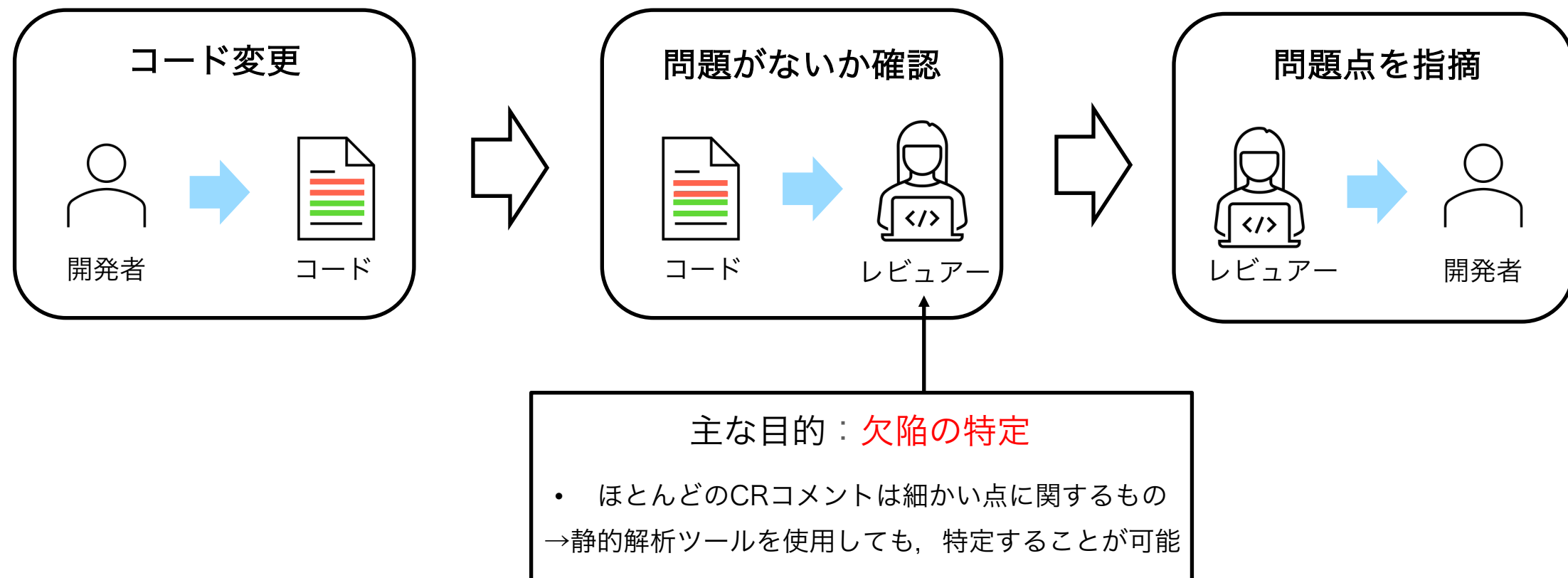
♣ *Bangladesh University of Engineering and Technology, Dhaka, Bangladesh*

2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM),
New Orleans, LA, USA, 2023 pp. 1-12.

コードレビュー (Code Review)

多くのソフトウェアプロジェクトで義務化されている

- 開発者は10-15%をCRに費やしている [2]



細かい（スタイルなど）CRが効率が悪い

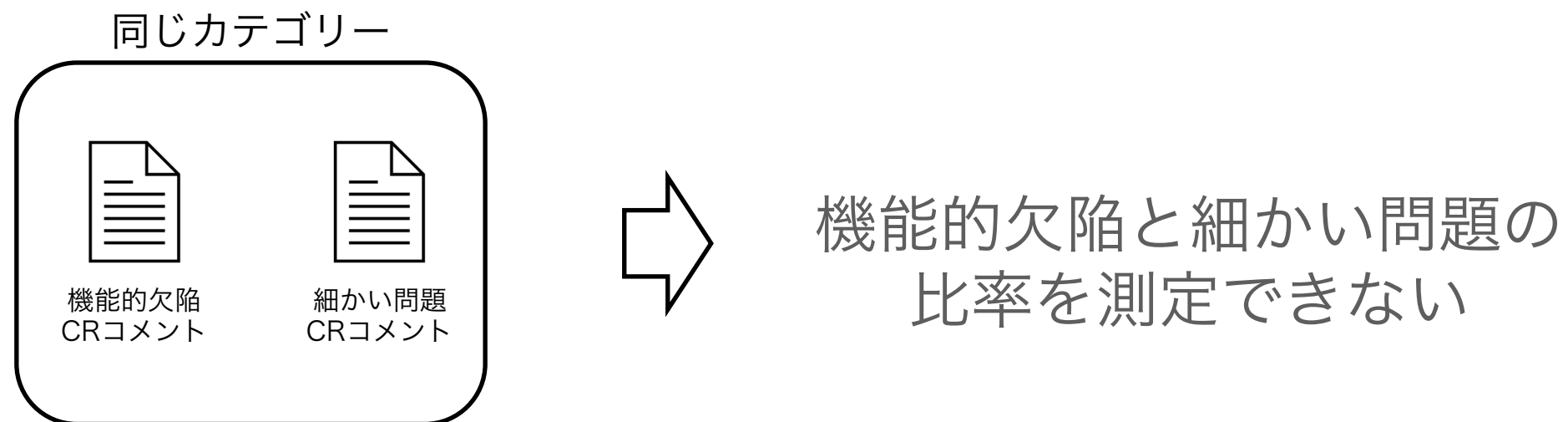
- CRの効果を向上させる必要がある

[2] A. Bosu, J. C. Carver, C. Bird, J. Orbeck, and C. Chockley, "Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at microsoft," IEEE Transactions on Software Engineering, vol. 43, no. 1, pp. 56-75, 2016.

問題点

多くの組織がCR分析を詳細に示している^[11-12]

- レビュー完了時間
- 優れたレビュー
- 特定された問題点の数 → 誤解を招く可能性がある



CRコメントを自動分類するモデルが必要

- 手動での分類は不可能であるため

[11] M. Hasan, A. Iqbal, M. R. U. Islam, A. I. Rahman, and A. Bosu, "Using a balanced scorecard to identify opportunities to improve code review effectiveness: An industrial experience report," Empirical Software Engineering, vol. 26, pp. 1–34, 2021.

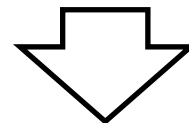
[12] C. Bird, T. Carnahan, and M. Greiler, "Lessons learned from building and deploying a code review analytics platform," in 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. IEEE, 2015, pp. 191–201.

研究の目的

Fregnanらは、CRを誘発させる変更を4つのカテゴリーに分類するモデルを開発^[13]

改善点

変更を引き起こさないCRコメントを考慮していない
使用していた機械学習アルゴリズムが古典的であった



目的とRQ

変更を引き起こさないCRコメントを考慮し、CodeBERTを使用したDNNモデルでのCRコメント自動分類器の開発

- RQ1：自動機械学習ベースのアプローチはCRコメントを分類できる？
- RQ2：開発したCRコメント分類ツールは最先端技術より優れている？

[13] E. Fregnan, F. Petruccio, L. Di Geronimo, and A. Bacchelli, "What happens in my code reviews? an investigation on automatically classifying review changes," Empirical Software Engineering, vol. 27, no. 4, p. 89, 2022

データセット

データ収集

- Open DevのGerritインスタンスから公開されている全てのCR
 - 2011年7月から2022年3月の期間で793,226件のCRを含む

手動でのラベリング

- Open Dev Novaプロジェクトから無作為に2500件を抽出
 - 投稿されたCRコメントが最も多い.

Group	Category	Description
Functional	Functional	Functional issues are defects where a code functionality is missing or implemented incorrectly. If requires large modification to resolve the issue.
	Logical	Logical issues are defects where there exist control flow problems or logical mistakes (wrong logic implementation, comparison issues).
	Validation	All types of user data sanitization issues or issues related to exception handling.
	Resource	Any kind of variable, memory, or file issues while handling or manipulating them.
	Timing	Any kind of synchronization issues while using the thread.
	Support issues	Any kind of support systems-related issues (e.g. configuration problem or version mismatch).
Refactoring	Interface	Any types of interfacing issues such as issues in an import statement, issues while interacting with the database or internal system.
	Solution approach	Review comments that suggest an alternative approach for problem-solving.
	Alternate Output	Review comments that address issues within the alert message, toast message, or error message.
	Code Organization	Code organization or refactoring issues presented in the catalog of Martin Fowler [44].
	Variable Naming	Review comments that address the violation of the variable naming convention.
Documentation	Visual Representation	Any kind of indentation, blank line, or code spacing-related issues.
	Documentation	Review comments that address issues related to code comments or documentation files for aiding code comprehension.
	Discussion	Comments that discuss design pattern or sourcecode architecture.
Discussion	Question	If reviewers ask anything to the code author for clarification.
	Praise	Review comments that praise or complement the developer.
False positive	False positive	An invalid concern. A CR comment is considered 'False Positive' if (a) the code owner explicitly mentions the comment as an invalid concern or (b) no subsequent change occurred nor the code owner agrees to a future change [8].

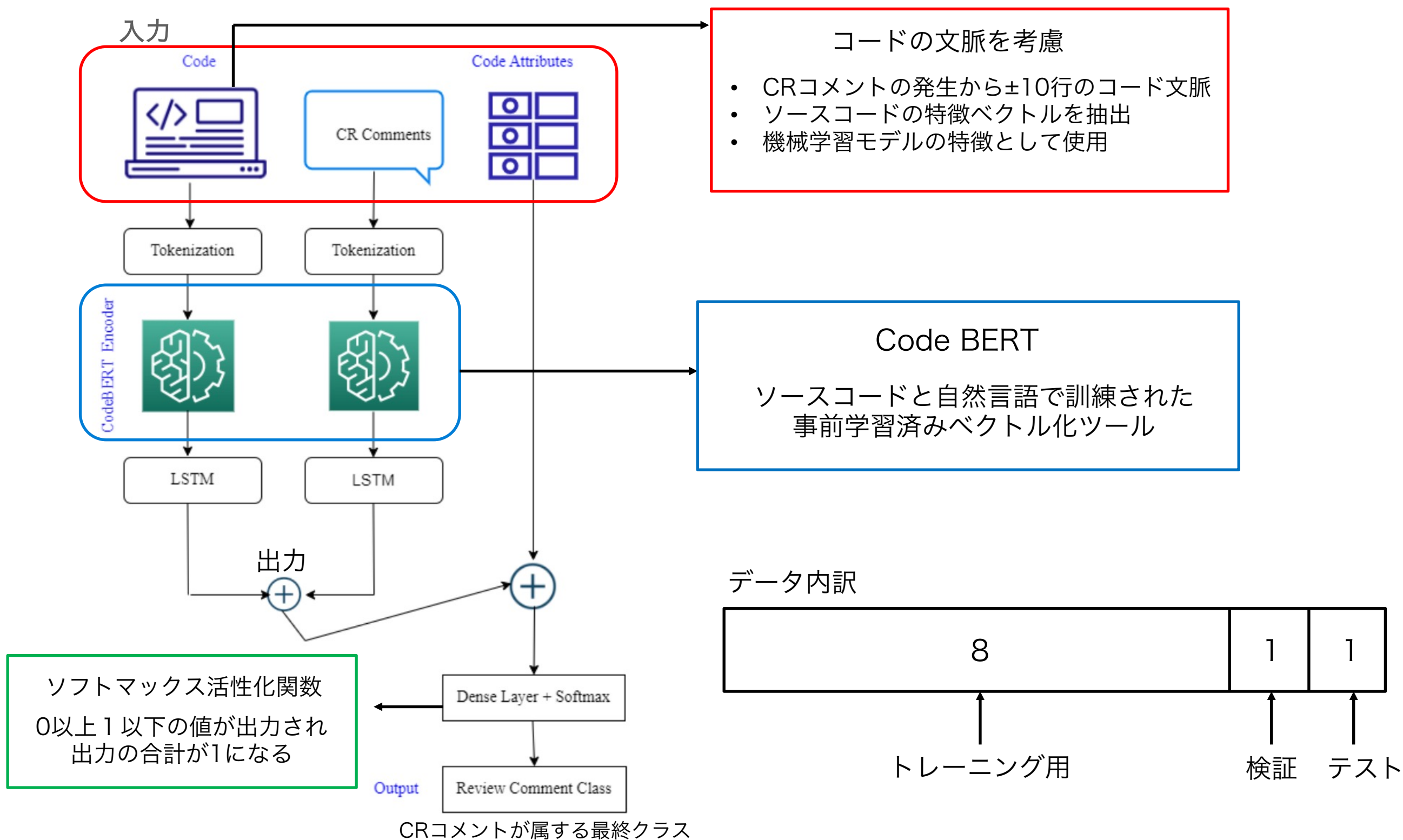


データセット

- 機能 : 13.13%
- リファクタリング : 32.71%
- ドキュメント : 21.17%
- 議論 : 24.34%
- 無効 : 8.64%

5つのグループに分ける

提案手法



評価指標

Precision (適合率)

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (再現率)

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score (F値)

$$F_1 \text{ score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Matthew相関係数 (MCC)

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

RQ1：自動機械学習ベースのアプローチはCRコメントを分類できる？

RQ1：結果

コード文脈，コードコメント，属性を個別に実験

Comment class	Code context only				Review comment only				Code attributes only			
	Precision	Recall	F1-Score	MCC	Precision	Recall	F1-Score	MCC	Precision	Recall	F1-Score	MCC
Discussion	0.326	0.208	0.232	0.396	0.586	0.706	0.637	0.543	0.049	0.121	0.062	0.183
Documentation	0.557	0.592	0.565	0.381	0.672	0.685	0.673	0.552	0.067	0.064	0.056	0.212
False Positive	0.033	0.013	0.019	0.414	0.441	0.194	0.257	0.571	0.009	0.036	0.014	0.235
Functional	0.176	0.055	0.056	0.414	0.416	0.327	0.337	0.568	0.068	0.297	0.093	0.224
Refactoring	0.415	0.719	0.523	0.283	0.602	0.624	0.604	0.520	0.255	0.452	0.263	0.145
Model Accuracy	0.418				0.579				0.230			

BERTとCodeBERTで比較

Comment class	BERT (tokenizer + vectorizer)				CodeBERT (tokenizer + vectorizer)			
	Precision	Recall	F1-Score	MCC	Precision	Recall	F1-Score	MCC
Discussion	0.516	0.570	0.522	0.449	0.611	0.685	0.638	0.547
Documentation	0.670	0.634	0.644	0.463	0.704	0.762	0.723	0.556
False Positive	0.124	0.041	0.056	0.487	0.376	0.158	0.207	0.576
Functional	0.380	0.116	0.162	0.483	0.408	0.358	0.365	0.572
Refactoring	0.461	0.632	0.523	0.409	0.628	0.633	0.616	0.527
Model Accuracy	0.502				0.593			

RQ2：結果

既存手法と提案手法の違い

- 既存手法はコードの属性に焦点を当てていた
- 提案手法はコードの文脈とCRコメントも考慮

⇒ 既存手法のモデルを再訓練し，比較

Comment class	Fregnan <i>et al.</i> 's Approach				Proposed Approach			
	Precision	Recall	F1-Score	MCC	Precision	Recall	F1-Score	MCC
Discussion	0.355	0.373	0.362	0.368	0.611	0.685	0.638	0.547
Documentation	0.477	0.497	0.483	0.382	0.704	0.762	0.723	0.556
False Positive	0.283	0.040	0.068	0.410	0.376	0.158	0.207	0.576
Functional	0.297	0.093	0.139	0.408	0.408	0.358	0.365	0.572
Refactoring	0.415	0.592	0.486	0.307	0.628	0.633	0.616	0.527
Model Accuracy	0.406				0.593			

提案手法は，既存手法よりもモデルの精度が優れていた

総括

コードレビューは多くのプロジェクトで行われている

- CRコメントは細かい問題（スタイル）などを示していることが多く，機能的な欠陥を分類することが必要

CodeBERTを使用したDNNモデルでのCRコメント自動分類器の開発

- コードの属性だけでなく，**コードの文脈**と**CRコメント**も考慮
- 既存のモデルよりも良い精度を表した

補足資料（ASTベースの属性）

Attributes	Description	Rationale
AST-based attributes		
anyInserted	Count of AST nodes inserted in the <i>destination</i> file.	Some code changes involve the addition of a few AST nodes. For example, in case of variable value assignment or logical changes.
anyDeleted	Count of AST nodes deleted in the <i>destination</i> file.	Some code changes involve the deletion of a few AST nodes. In case of variable value assignment or logical changes, few AST node deletions might occur.
getMovedSrcs	How many AST nodes moved from the <i>source</i> file.	If the <i>source</i> code chunk is moved to the <i>destination</i> , it might indicate a refactoring change.
updatedSrcs	How many <i>sources</i> AST nodes are updated in the <i>destination</i> file?	Update of a few numbers of AST nodes may indicate logical changes.
anythingInLineMoved	Count of <i>source</i> AST nodes that were within the Review Comment Range (RCR) but moved elsewhere in the <i>destination</i> file.	Prior studies suggest most changes occur in close proximity to the review comment. The number of AST nodes moved within close proximity can provide valuable information about the change.
anythingInLineUpdated	Count of <i>source</i> AST nodes that were within the RCR and were updated in the <i>destination</i> file.	Similarly, the number of AST nodes updated within close proximity of the review comment can provide valuable information about the change.
anythingInLineDeleted	Count of <i>source</i> AST nodes that were within the RCR and were deleted.	Similarly, we argue that the number of AST nodes deleted within close proximity of the review comment can provide valuable information about the change.
anythingMovedIntoLine	If an AST node is in the RCR of the <i>source</i> , then the number of child nodes moved within that node in the <i>destination</i> file.	Number of nodes moved within RCR can provide crucial information for review comment classification, it can be indicative of refactoring changes.
anythingInsertedIntoLine	If an AST node is in the RCR of the <i>source</i> , then the number of child nodes that are newly inserted within that node in the <i>destination</i> file.	Number of nodes newly inserted within RCR can provide crucial information for review comment classification, it can be indicative of larger changes.

補足資料（変更・修正ベースの属性）

Change-based attributes		
insertedIfConditions	How many if statements in the <i>destination</i> file were inserted?	Insertion of 'if' condition indicates logical Functional change.
deletedIfConditions	How many if statements in the <i>source</i> file were deleted?	Similarly, deletion of the 'if' condition indicates logical Functional change.
elseInserted	How many else inserted in the <i>destination</i> file?	Insertion of 'else' condition indicates control flow change, which is a Functional change.
elseDeleted	How many else statements in the <i>source</i> file were deleted?	Similarly, deletion of the 'else' condition indicates logical Functional change.
entireLineMoved	How many full lines in the RCR were moved in the <i>destination</i> file?	Number of lines moved within close proximity can be valuable, for example, a large number of moved lines can indicate a structural rearrangement.
entireLineDeleted	How many full lines in the RCR were deleted from the <i>source</i> file?	Similarly, a large number of deleted lines can indicate a structural change or a larger change.
stringsUpdated	Number of strings in the <i>source</i> file that was updated.	Higher number of strings updated in the source file can indicate an 'Alternate Output' change.
magicStringsReplaced	Number of Magic strings replaced with variables.	Magic Strings are strings that are specified internally and impact the code behavior. Magic string replacement may indicate control flow-related change.
movedBlocksInIfConditions	Number of blocks within the if conditions were moved in the <i>destination</i> file.	Number of moved blocks within the 'if' condition indicates control flow-related issues within code.
insertedAssertConditions	Number of asserts inserted in <i>destination</i> file	As assert check for condition, insertion of assert condition might indicate functional change.
insertedTryCatch	Number of try-catch nodes inserted in the <i>destination</i> file.	Inserted try/catch statement may indicate Validation issues within code.
removedTryCatch	Number of try-catch nodes in the <i>source</i> file that was removed in the <i>destination</i> file.	Removed try/catch statement may indicate Validation issues or even a Structural change within the code.
updatedValueAssignments	Number of updated statements in the <i>destination</i> file where any variables' value was updated.	Variables' value update can indicate logical functional issues.
updatedFunctionArguments	Number of function arguments that were updated in the <i>destination</i> file.	Number of updated arguments within a function can provide valuable information about the change.
File-based attributes		
hasNewFile	A binary variable indicating <i>destination</i> file existence.	If 0, then it indicates no change occurred, so the comment might be false positive.
hasOldFile	A binary variable indicating <i>source</i> files' existence.	Similarly to the previous one, this variable's value can provide information about a comment.
cyclomaticComplexity	The Cyclomatic Complexity proposed by McCabe [47] of the <i>source</i> file.	Cyclomatic complexity can provide vital information regarding the issue of the code or the types of review comments.
commentLOC	The line number of the <i>source</i> file where the review comment was made.	Comment line number can provide crucial information, such as if the commentLOC is small, then most probably the review comment is targeted for import statements.