

Code Comment Inconsistency Detection and Rectification Using a Large Language Model

Guoping Rong
Nanjing University
Nanjing, China
ronggp@nju.edu.cn

Yongda Yu*
Nanjing University
Nanjing, China
yuyongda@smail.nju.edu.cn

Song Liu
Nanjing University
Nanjing, China
522023320112@smail.nju.edu.cn

Xin Tan
Nanjing University
Nanjing, China
522023320141@smail.nju.edu.cn

Tianyi Zhang
Nanjing University
Nanjing, China
522023320208@smail.nju.edu.cn

Haifeng Shen
Southern Cross University
Gold Coast, Australia
haifeng.shen@scu.edu.au

Jidong Hu*
Zhongxing Telecom Equipment
Xi'an, China
hi.jidong@zte.com.cn

背景

コードコメントの一貫性

- コードコメント

コード理解を助けるために使用される（例 メソッドの目的と戻り値を記述）

コードとコメントの内容が一致していない場合がある

- 不一致の原因

- 開発者のスキルや時間の制約
- コード変更時のコメント更新忘れ

- 不一致による影響

- コードの理解しやすさ，保守性が低下する
- 外部のライブラリの機能について誤解を生む
- コメントへの信頼が低下する

先行研究

不一致の検出

- ルールベース

事前定義されたルールやパターンを使用する

- 学習ベース

機械学習, 深層学習技術を使用する

コードとコメントの内容の不一致に対処するには検出だけでなく
その修正も必要であるが, **不一致の修正に着目した研究は少ない**

目的・アプローチ

不一致の検出と修正を両方行う手法を提案する

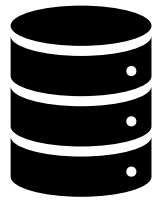
2つのモードで検出，修正を行う

- post hoc（事後型）
- just-in-time（即時型）

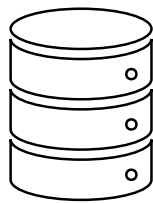
どちらにも対応するように手法を構築

提案手法 | 概略

(1) FT用のデータセット作成



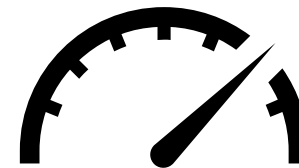
事後型



即時型

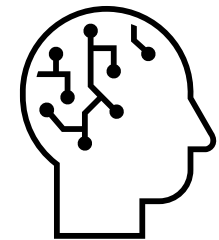
2つのモードで
データセットを作成

(2) ファインチューニング



損失関数, FT手法を工夫

(3) 検出・修正



2つのモードで
タスクの実行

提案手法 | FT用データセット作成①

データセット

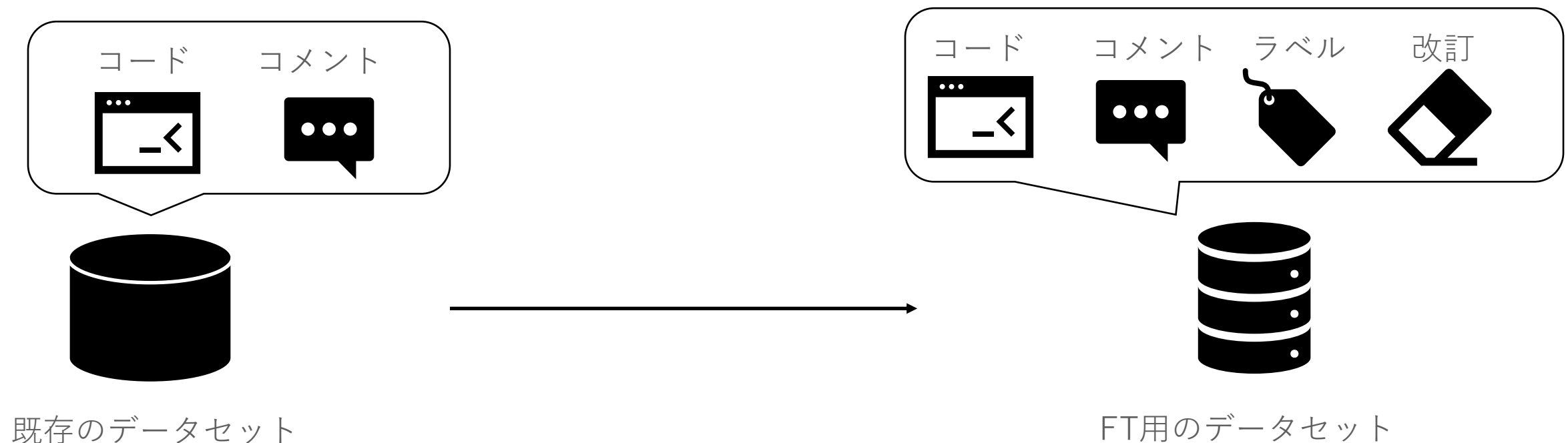
既存研究で使用されているもの

言語 : Java

構成要素 : 連続して提出されたコードとコメントのペア (C_1, M_1) (C_2, M_2)

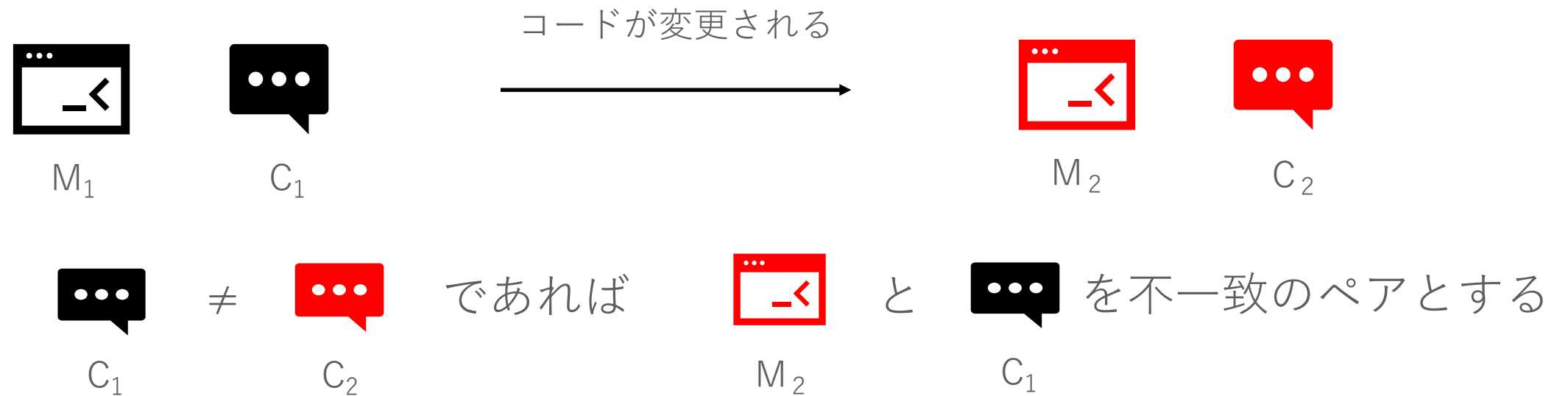
C:コメント M:コード

@return @param Summaryなどのコメント要素が含まれる



提案手法 | FT用データセット作成②

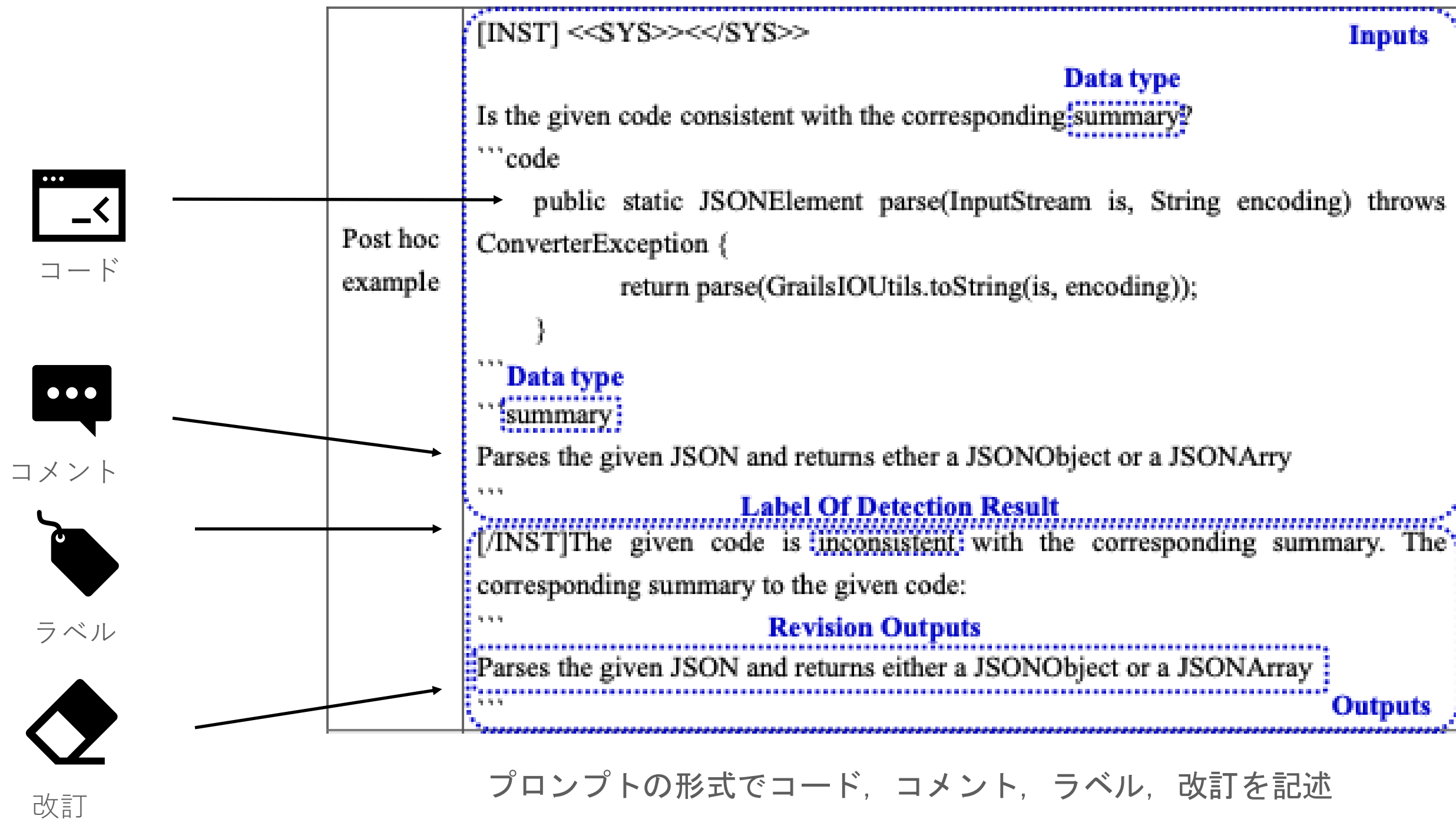
- ラベル（不一致かどうか）



- 改訂（コメントの正解）



提案手法 | FT用データセット作成③



提案手法 | ファインチューニング①

使用モデル

- CodeLLaMA

損失関数(独自に作成)

$$L_{ID} = \alpha L_I + (1 - \alpha) \sum_{i=1}^n \log P(x_i | x_{<i})$$

L_I : 分類タスクの損失関数 第2項 : 標準的な言語モデルの損失関数

α : 任意の値 (通常は0.5)

$$L_I = \log((1 - \varepsilon)P(I|Ins, C, M) + \frac{\varepsilon}{K})$$

$P(I|Ins, C, M)$: 正解ラベルに対するモデルの予測確率

ラベルスムージングが利用されている

→ノイズやラベル付エラーのあるデータに対して強くなる

提案手法 | ファインチューニング②

ファインチューニング手法

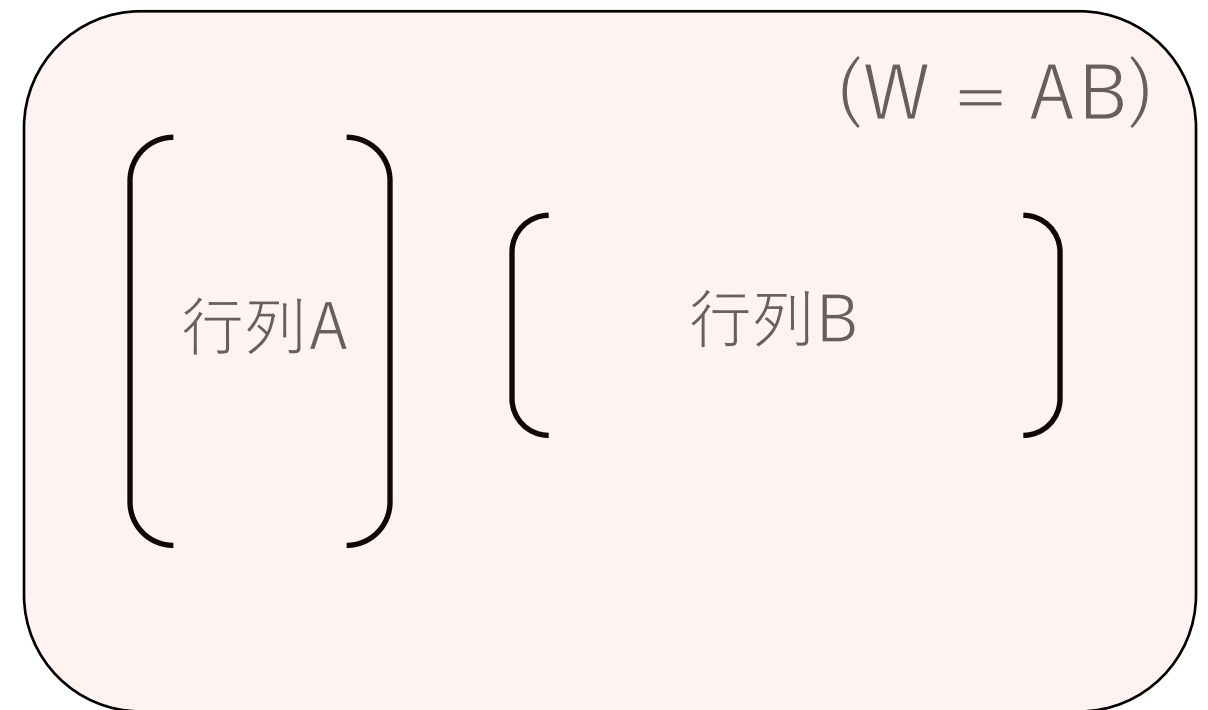
- LoRA (Low-Rank Adaptation)

調整するパラメータを少なくする方法（元の1,000分の1から10,000分の1）



パラメータの更新をしない

+



パラメータの更新をする

評価実験

リサーチクエスチョン

- 1: 提案手法は不一致**検出**タスクでどのようなパフォーマンスを示すか
- 2: 提案手法は不一致**修正**タスクでどのようなパフォーマンスを示すか

評価指標

- RQ1

Precision / Recall / F1 / Accuracy

- RQ2

xMatch / BLUE-4 / GLEU / Meteor / SARI

結果 | RQ1

事後型

Approach	Precision	Recall	F1	Accuracy
CodeBert BOW	68.9	73.2	70.7	69.8
SEQ	60.6	73.4	66.3	62.8
GRAPH	62.6	72.6	67.2	64.6
HYBRID	56.3	80.8	66.3	58.9
BERT	72.1	71.9	72.0	72.1
Longformer	92.7	81.0	86.4	87.3
DocChecker	-	-	74.3	72.3
gpt-3.5-turbo-1106	62.6	62.0	62.3	62.5
gpt-4-0125-preview	59.2	81.7	68.7	62.6
CodeLLaMA-7B	94.0	84.6	89.0	89.6

即時型

- 差分の表現方法で結果が変わる

<DELETE>,<INSERT>などのタグ

よりも +や-で表現する方が良い

既存手法よりも精度が向上

Approach	Precision	Recall	F1	Accuracy
CodeBERT BOW	67.4	76.8	71.6	69.6
SEQ	80.7	73.8	77.1	78.0
GRAPH	79.8	74.4	76.9	77.6
HYBRID	80.9	74.7	77.7	78.5
SEQ + features [†]	88.4	73.2	80.0	81.8
GRAPH + features [†]	83.8	78.3	80.9	81.5
HYBRID + features [†]	88.6	72.4	79.6	81.5
BERT	76.4	78.3	77.3	77.1
Longformer	80.3	76.5	78.3	78.8
gpt-3.5-turbo-1106	50.0	100.0	66.7	50.0
gpt-4-0125-preview	51.5	98.8	66.7	52.8
C4RLLaMA	85.8	82.6	84.1	84.5
C4RLLaMA with standard Diff*	95.9	87.3	91.4	91.8

結果 | RQ2

Approach	xMatch	BLEU-4	GLEU	SARI	Meteor
Jointly trained SEQ	62.3	76.6	75.6	42.0	75.9
Jointly trained GRAPH	59.4	76.6	75.8	42.5	75.1
Jointly trained HYBRID	62.3	76.9	75.9	42.3	75.6
Pretrained SEQ	61.4	77.0	76.2	42.4	75.6
Pretrained GRAPH	60.8	76.6	75.8	41.8	74.9
Pretrained HYBRID	61.6	77.2	76.4	42.3	75.8
<i>C4RLLaMA</i> Post hoc	50.0	81.7	82.0	85.2	88.5
<i>C4RLLaMA</i> <i>just-in-time</i>	50.0	77.5	77.5	82.0	86.8
<i>C4RLLaMA</i> <i>just-in-time</i> with standard ‘Diff’ [*]	50.0	82.6	83.0	86.7	89.2

差分の表現を変更した即時型の提案手法が最も良い性能を示した

人間評価では

事後型 : 65.0% 即時型 : 55.9%が修正に成功

課題点

- スペルミスなどは修正できたが、意味的な内容は不正確
- モデルサイズにより、ユーザの意図を理解する上での限界がある

議論

提案手法の優位性の背景

- LLMを利用したこと，損失関数の設計，FTを工夫したことにより複雑なタスクを実行することができた

不一致問題解決の意義

- 大規模言語モデル（特にコード生成モデル）のトレーニングにはコードが使われているため，不一致コメントが学習性能を下げる可能性がある

改善点

- より大きなモデルを使用することで性能向上が期待できる
- 多様な形式や複雑さのコメントへの対応が必要

総括

コードとコメントの内容の不一致を検出・修正する手法を提案

- FT用のデータセット作成, 損失関数の設計, FT手法を工夫

検出・修正において既存手法よりも精度が向上

- 人間評価により, 修正が実際の開発環境で有用であることが示された

複雑な内容のコメントの修正には課題が残った

