

Improving Retrieval-Augmented Code Comment Generation by Retrieving for Generation

Hanzhen Lu[†], Zhongxin Liu^{*†‡}

[†]The State Key Laboratory of Blockchain and Data Security, Zhejiang University, China

[‡]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security, China
{luhanzhen, liu_zx}@zju.edu.cn

Proceedings of the 40th IEEE International Conference on Software Maintenance and Evolution (ICSME 2024)

背景

- ソフトウェアの規模は拡大し続けており，コードの理解と保守に時間が多大な時間が費やされる [1, 2]
- コードコメントはコード理解に重要
- コード変更時に手作業でコメントを維持するのは困難 [3, 4]

⇒高品質なコメントを自動生成する研究が提案

[1] : J. A. J. Ko, B. A. Myers, M. J. Coblenz, and H. H. Aung, "An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks," *IEEE Transactions on software engineering*, vol. 32, no. 12, pp. 971–987, 2006.

[2] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li, "Measuring program comprehension: A large-scale field study with professionals," *IEEE Transactions on Software Engineering*, vol. 44, no. 10, pp. 951–976, 2017.

[3] : S. C. B. de Souza, N. Anquetil, and K. M. de Oliveira, "A study of the documentation essential to software maintenance," in *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information*, 2005, pp. 68–75.

[4] M. Kajko-Mattsson, "A survey of documentation practice within corrective maintenance," *Empirical Software Engineering*, vol. 10, pp. 31–55, 2005.

先行研究

- 情報検索(IR)ベース^[5, 6] : BM25, Dense Retrievers
 - 過去のコードとコメントのペア **模範解答** から, 類似のコードを検索し対応するコメントを生成コメントとする
 - 生成コメントとコードに意味のずれが生じる場合あり
- ニューラルネットワーク(NN)ベース^[7, 8, 9] :
 - コメント生成を言語翻訳タスクとみなして生成
 - IRベースではできた低頻度のトークン生成ができない

[5] : E. Wong, J. Yang, and L. Tan, "Autocomment: Mining question and answer sites for automatic comment generation," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2013, pp. 562–567.

[6] : E. Wong, T. Liu, and L. Tan, "Qocom: Mining existing source code for automatic comment generation," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, IEEE, 2015, pp. 380–389.

[7] : S. Iyer, I. Korstas, A. Cheung, and L. Zettlemoyer, "Summarizing source code using a neural attention model," in *54th Annual Meeting of the Association for Computational Linguistics* 2016. Association for Computational Linguistics, 2016, pp. 2073–2083.

[8] : X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, "Deep code comment generation," in *Proceedings of the 26th conference on program comprehension*, 2018, pp. 200–210.

[9] : X. Hu, G. Li, X. Xia, D. Lo, S. Lu, and Z. Jin, "Summarizing source code with transferred api knowledge," 2018.

先行研究

- IR + NN ベース (Retrieval-Augmented Comment Generation) [10, 11, 12, 13]
 - 元のコードに模範解答を加えて入力し, 新たなコメントを生成
 - 従来のRACGでは検索器と生成器が独立して構築され, 必ずしも生成に最も役立つ模範解答が選ばれない

目標コードコメント

Code 1:

```
'change private-browsing config to true and emit signal'
def test_cache_config_enable_private_browsing(
    config_stub, tmpdir):
    config_stub.data = {'storage': {'cache-size':
        1024}, 'general': {'private-browsing':
        False}}
    disk_cache = cache.DiskCache(str(tmpdir))
    assert (disk_cache.cacheSize() == 0)
    preload_cache(disk_cache)
    assert (disk_cache.cacheSize() > 0)
    config_stub.set('general', 'private-browsing',
        True)
    assert (disk_cache.cacheSize() == 0)
```

従来の検索器の 模範解答

Code 2:

```
'test if cache is empty after clearing it'
def test_cache_clear_activated(config_stub, tmpdir):
    config_stub.data = {'storage': {'cache-size':
        1024}, 'general': {'private-browsing':
        False}}
    disk_cache = cache.DiskCache(str(tmpdir))
    assert (disk_cache.cacheSize() == 0)
    preload_cache(disk_cache)
    assert (disk_cache.cacheSize() != 0)
    disk_cache.clear()
    assert (disk_cache.cacheSize() == 0)
```

最も目標コメントに 近くなる模範解答

Code 3:

```
'change private-browsing config to false and emit signal'
def test_cache_config_disable_private_browsing(
    config_stub, tmpdir):
    config_stub.data = {'storage': {'cache-size':
        1024}, 'general': {'private-browsing':
        True}}
    url = 'http://qutebrowser.org'
    metadata = QNetworkCacheMetaData()
    metadata.setUrl(QUrl(url))
    assert metadata.isValid()
    disk_cache = cache.DiskCache(str(tmpdir))
    assert (disk_cache.prepare(metadata) is None)
    config_stub.set('general', 'private-browsing',
        False)
    content = 'cute'
    preload_cache(disk_cache, url, content)
    assert (disk_cache.data(QUrl(url)).readAll()
        == content)
```

[10] : B. Wei, Y. Li, G. Li, X. Xia, and Z. Jin, "Retrieve and refine: exemplar-based neural comment generation," in Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, 2020, pp. 349–360.

[11] : J. Zhang, X. Wang, H. Zhang, H. Sun, and X. Liu, "Retrieval-based neural source code summarization," in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, 2020, pp. 1385–1397.

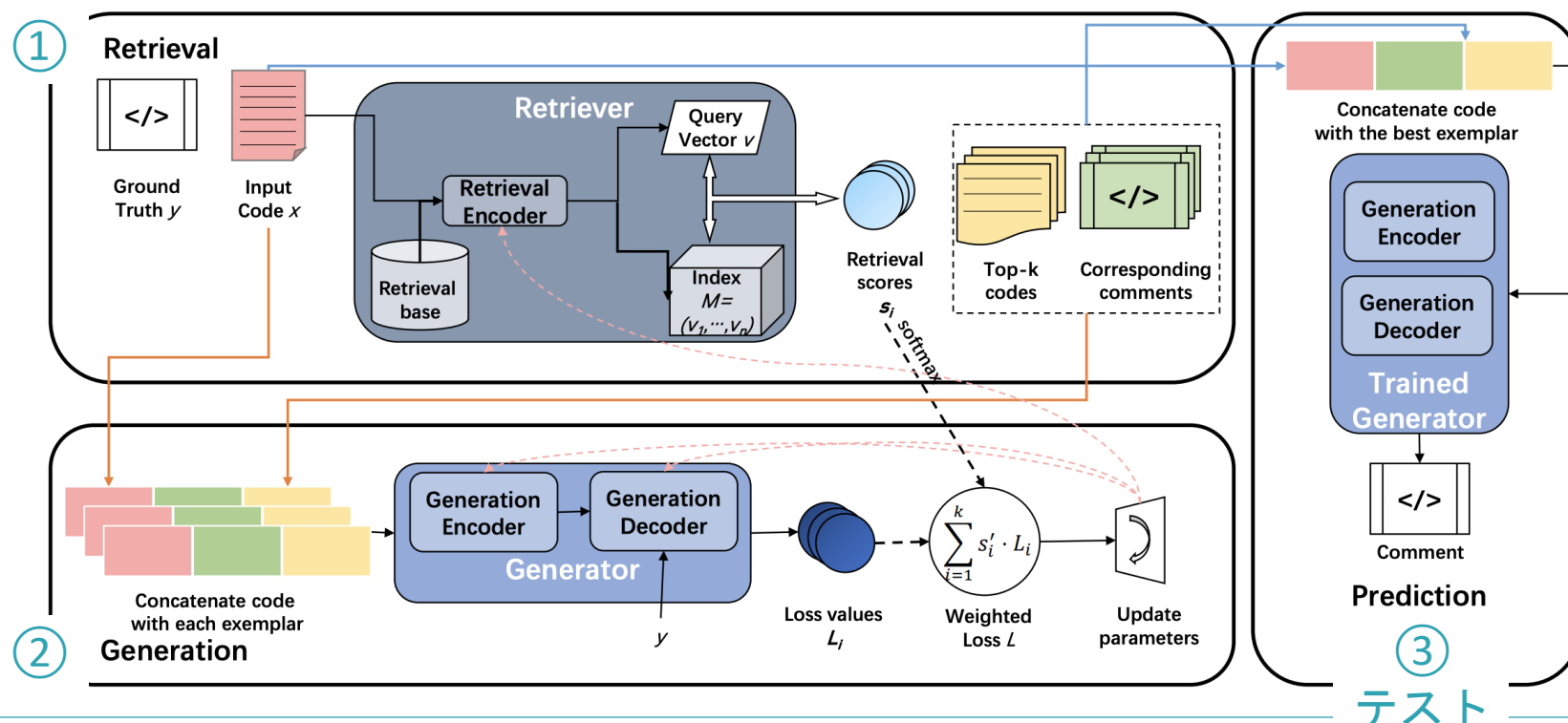
[12] : J. Li, Y. Li, G. Li, X. Hu, X. Xia, and Z. Jin, "Editsum: A retrieve-and-edit framework for source code summarization," in 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, 2021, pp. 155–166.

[13] : F. Mu, X. Chen, L. Shi, S. Wang, and Q. Wang, "Automatic comment generation via multi-pass deliberation," in Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, 2022, pp. 1–12.

提案手法

JOINTCOM

- 検索器と生成器を共同で学習することで，生成に最適な模範解答を選び，高品質なコメントを生成する



JOINTCOM : 訓練プロセス

① 検索 : ニューラルエンコーダ(CodeT5)への入力コード
に対し模範解答をデータベースから検索

1. 検索対象のコードスニペットをベクトル化してインデックスとして格納
2. 訓練時に入力コードに対して類似度スコア_※を計算しインデックスから上位k件の模範解答を検索

※コサイン類似度やL2ノルムなど

JOINTCOM : 訓練プロセス

② 生成 : k個の各模範解答と入力コードを組み合わせて コメント生成

1. 各模範解答と入力コードのペアごとについて
コメントを生成
2. 生成したコメントと正解コメントを比較し
各模範解答ごとに生成損失 $\mathcal{L}_i$ を計算

$$\mathcal{L}_i = -\log p(y|x, r_i) = -\sum_{t=1}^{|y|} \log p(y_t|y_{<t}, x'_i)$$

JOINTCOM : 訓練プロセス

③ 訓練 : 検索器と生成器を共同で学習

1. 各生成損失 L_i をsoftmax関数で正規化

$$(s'_1, s'_2, \dots, s'_n) = \text{softmax}(s_1, s_2, \dots, s_n)$$

2. 各生成損失 L_i を類似度スコアで重み付けして合計し、最終的な生成損失 L とする

$$\mathcal{L} = \sum_{i=1}^n \mathcal{L}_i \cdot s'_i$$

3. 生成損失 L が最小となるよう検索器と生成器を最適化

JOINTCOM : テストプロセス

- 入力コードに対して最も関連性の高い模範解答1件からコメントを生成
- 評価指標
 - コーパスレベルBLEU : 全体の生成結果における一致度を評価
 - センテンスレベルBLEU : 各生成コメントで独立して一致度を評価
 - ROUGE-L : 生成コメントと元コメント間の最長共通部分を評価
 - METEOR : 単語ごとの一致度に加え語形の類似性や同義語も評価
 - CIDEr : 生成コメントと元コメントの一致度を単語出現頻度で重み付けして評価

データセット

- コード要約モデル評価に広く使われるデータセット
 - JCSD : GitHub上の9,714件のJavaプロジェクトから収集された85,000件の関数とコメントのペア
 - 関数の最初のJavadoc文を要約コメントとする
 - PCSD : GitHubから収集されたPythonの関数とコメントのペア

TABLE II
STATISTIC OF DATASETS

Dataset	JCSD	PCSD
train	69708	55538
validation	8714	18505
test	6489	18142
Avg. tokens in code	99.9	86.7
Avg. tokens in comment	17.1	9.4
Max. tokens in code	4842	3339
Max. tokens in comment	670	50

データセットの前処理

- JCSD

1. javalangを用いてトークン化
2. CamelCaseやsnake_caseなどの複合トークンをサブトークンに分割
3. 訓練とテストセット両方に出現するコードはテストセットから削除

- PCSD

1. 訓練とテストセットの重複が確認されたためGaoらの提供するデータセットを使用して重複サンプルを除去
2. tokenizeパッケージを使用して各コードスニペットをトークン化
3. 前処理されたコードが訓練セットに存在するテストセット内のデータを削除

結果

RQ1 : JOINTCOMはベースラインと比較して
どの程度効果的か？

	Method	JCS					PCSD				
		C-BLEU	S-BLEU	ROUGE-L	METEOR	CIDEr	C-BLEU	S-BLEU	ROUGE-L	METEOR	CIDEr
検索 ベース	LSI	22.5	18.2	35.0	16.1	1.911	21.7	16.0	39.2	16.7	1.850
	VSM	23.3	19.1	36.6	17.0	2.038	23.1	17.1	41.1	18.0	2.035
	NNGen	24.2	19.8	37.5	17.4	2.092	24.3	18.1	42.4	18.6	2.108
NMT ベース	Hybrid-DRL	-	13.3	26.5	13.5	1.656	12.8	8.2	39.2	14.9	1.304
	SG-Trans	22.9	20.9	42.0	18.4	2.257	25.3	18.8	47.0	20.6	2.349
検索＋生成	Rencos	-	20.6	42.0	17.3	2.209	25.8	19.0	46.4	20.3	2.301
	EditSum	-	16.9	38.6	15.2	1.865	15.9	11.7	37.3	14.2	1.476
	DECOM	24.9	22.3	43.9	19.4	2.398	27.1	20.0	48.7	21.8	2.458
事前学習	CodeT5	15.4	15.3	42.1	16.5	1.989	22.4	16.3	49.0	21.6	2.346
	Unixcoder	14.9	19.4	43.0	16.5	2.345	22.7	18.1	49.1	20.7	2.482
	CodeT5+	20.3	19.4	44.3	18.6	2.332	26.8	20.3	51.3	23.3	2.710
JOINTCOM		26.8	27.4	51.1	22.5	3.080	33.2	26.4	56.2	26.6	3.307

- 5つの指標すべてでベースラインを大幅に上回った

結果

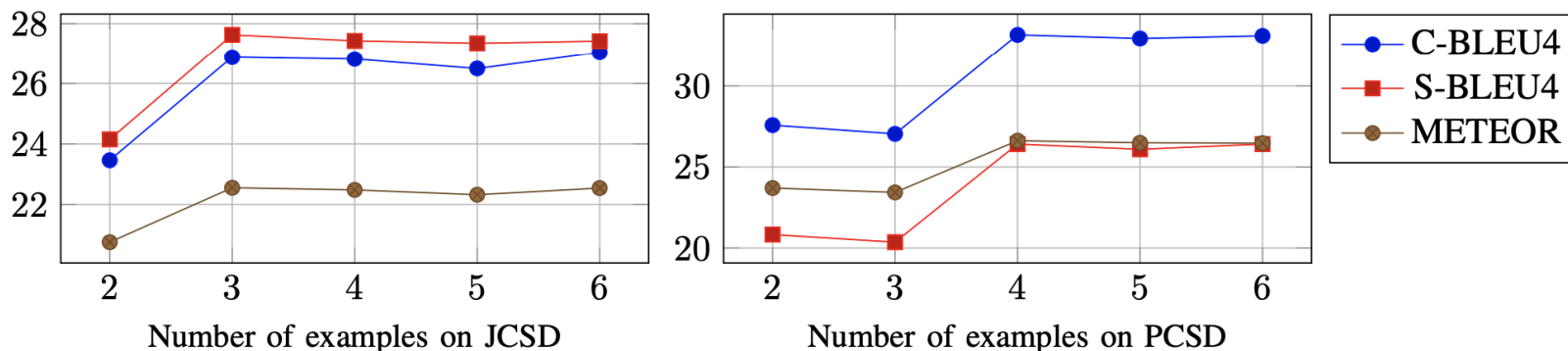
RQ2 : 学習手法はJOINTCOMにどのような利益をもたらすのか？

Retrieval	Generation	PCSD				
		C-BLEU	S-BLEU	ROUGE-L	METEOR	CIDEr
RAF _{BM25}		31.5	24.4	54.7	25.8	3.110
RAF _{Trained Encoder}		27.3	20.3	51.6	23.7	2.723
JOINTCOM	RAF _{BM25}	32.8	26.0	55.5	26.3	3.249
RAF _{BM25}	JOINTCOM	31.1	24.3	54.7	25.6	3.109
Random	RAF _{BM25}	11.5	6.3	41.5	17.1	1.393
Random	JOINTCOM	13.9	8.7	44.3	18.3	1.679
JOINTCOM		33.2	26.4	56.2	26.6	3.307

- JOINTCOMの学習手法は有効

結果

RQ3 : 学習中に利用される模範解答の数は
JOINTCOMの有効性にどんな影響を与えるか？



- k=4以降で収束する

結果

人間による評価

- 博士と修士の学生4人ずつ計8人の参加者
- データセットから100件取り出し4つのグループに分け、2人の参加者にそれぞれ評価させる

Approach	JCSD			PCSD		
	Naturalness	Informativeness	Usefulness	Naturalness	Informativeness	Usefulness
CodeT5+	$4.52 \pm 0.76(0.189)$	$3.52 \pm 0.83(<0.05)$	$3.74 \pm 1.00(<0.05)$	$4.36 \pm 0.94(0.295)$	$3.52 \pm 0.88(<0.05)$	$3.61 \pm 1.08(<0.05)$
DECOM	$4.59 \pm 0.71(0.409)$	$3.55 \pm 0.98(<0.05)$	$3.71 \pm 1.04(<0.05)$	$4.52 \pm 0.78(0.831)$	$3.53 \pm 0.97(<0.05)$	$3.54 \pm 1.15(<0.05)$
JOINTCOM	4.63 ± 0.68	3.76 ± 0.87	3.95 ± 0.99	4.56 ± 0.85	3.82 ± 0.85	3.91 ± 1.06

* The number in parentheses is the p-value in the Wilcoxon signed-rank test.

- 自然さ，情報量，有用性の3つ全ての観点で，ウィルコクソンの符号順位和検定でベースラインとの有意差を確認

議論：質的分析

- JOINTCOMの性能が劣るケース
- 適切な模範解答が見つからなかった場合
- 模範解答の質が低い場合
- 専門知識などが必要な場合
- データセットの偏りの影響

EXAMPLE OF QUALITATIVE ANALYSIS-3

Code:

```
static String toLowerCase(String s) {  
    int len = s.length();  
    StringBuilder sb = null;  
    for (int i = 0; i < len; i++) {  
        char c = s.charAt(i);  
        if (c >= 'a' && c <= 'z' || c == '.' || c == '-') {  
            if (sb != null) sb.append(c);  
        } else if (c >= '0' && c <= '9' || c == '-') {  
            if (sb != null) sb.append(c);  
        } else if (c >= 'A' && c <= 'Z') {  
            if (sb == null) {  
                sb = new StringBuilder(len);  
                sb.append(s, 0, i);  
            }  
            sb.append((char) (c - CASE_DIFF));  
        } else {  
            throw new IllegalArgumentException("Invalid  
                characters in hostname");  
        }  
    }  
    return sb == null ? s : sb.toString();  
}
```

Comment:

convert to lower case, and check that all chars are ascii alphanumeric, '-' or '.' only.

Retrieved comment

JOINTCOM: converts all of the characters in the string to upper case, based on the locale.

Prediction

JOINTCOM: converts all of the characters in the string to lower-case.

まとめと所感

- JOINTCOMは従来手法と比較して生成コメントの品質を大幅に向上させた
- 今後の課題
 - 他のプログラミング言語やバグ修正コード変換などのコード関連タスクへの適用など

所感

- 全体を把握しやすかったが詳細の構成で読みにくかった