

# Towards Automating Code Review Activities

Rosalia Tufano, Luca Pascarella, Michele Tufano,  
Denys Poshyvanyk, Gabriele Bavota

43rd International Conference on Software  
Engineering (ICSE 2021)

発表者:松尾 和輝

# 背景と目的

- 背景

- コードレビューは産業界とオープンソースプロジェクトの両方で普及している
- レビューされたコードはバグが発生する可能性が低く、理解や保守が容易である
- コードレビューを手動で行うことは、多くのコストがかかる

- 目的

- コードレビュープロセスの一部を自動化するモデルの提案
  - コードとレビューからコードを修正する部分の自動化

# アプローチ

- データ収集, データ成形, 機械学習によりモデルを構築し, 性能を評価する

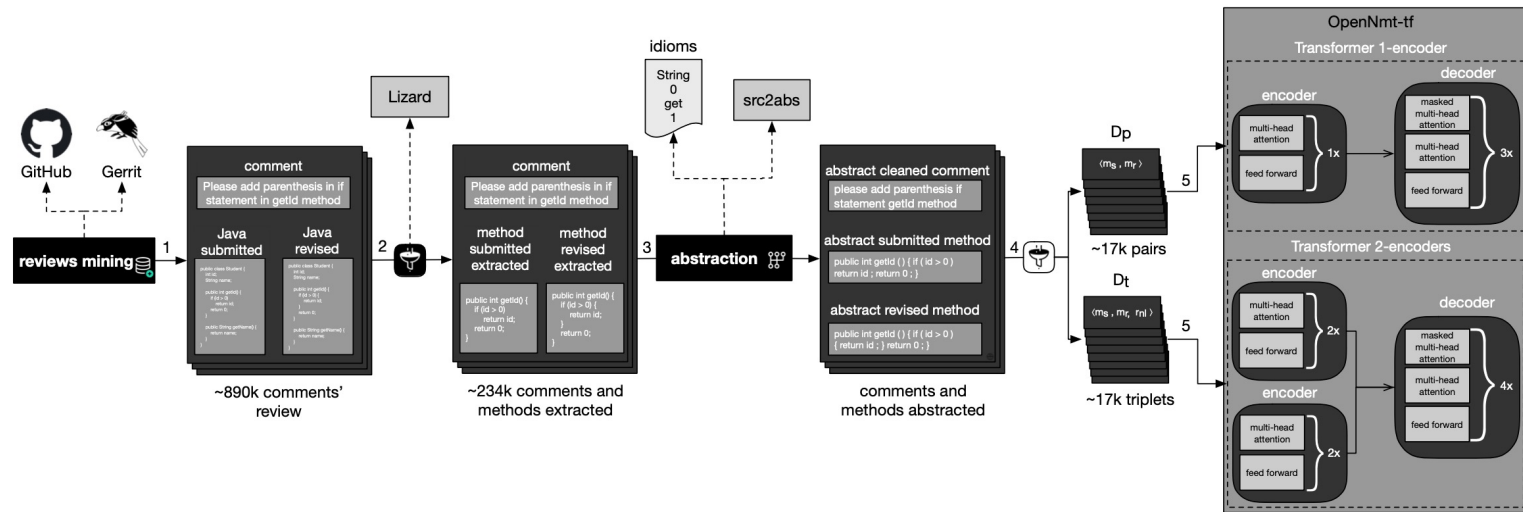
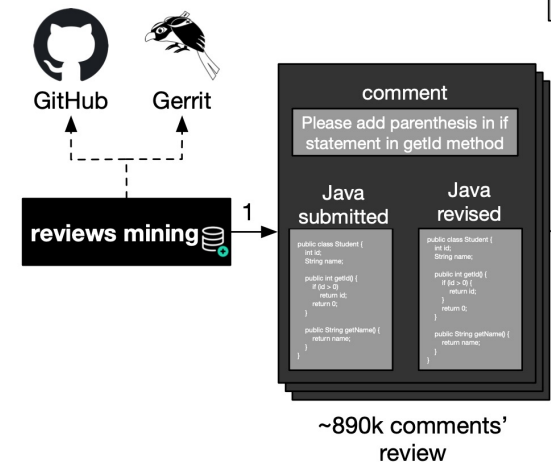


Fig. 1: Approach overview.

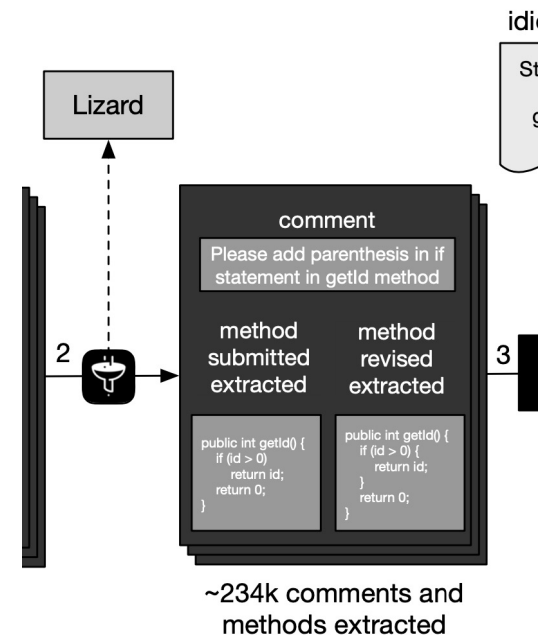
# データ収集

- GithubやGerritから，レビュー用に提出されたコードms，レビューrnl，修正されたコードmrをトリプレットとして収集する．
- Gerritからは6388個のjavaプロジェクト，Githubからは少なくとも50件のPRを持つ2566個のGitHub Javaリポジトリからトリプレットを収集する



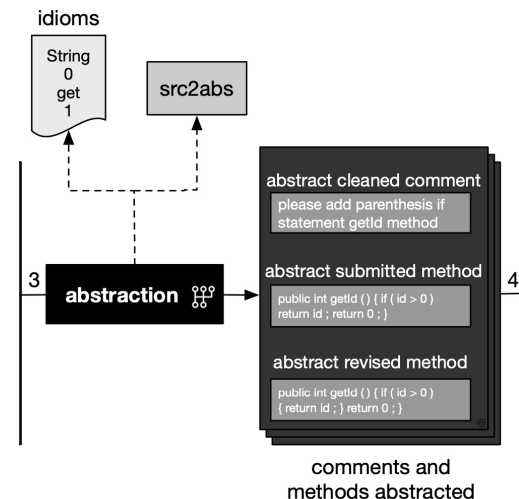
# データ成形

- PythonライブラリであるLizardを利用して、トリプレットのmsとmrをそれぞれ解析する
- 解析により、msとmrのそれぞれでレビューに関するメソッドを抽出する



# データ成形

- **src2abs**というツールを用いて、**ms**と**mr**とレビューの抽象化を行う
- 抽象化の際にエラーが発生したトリプレットをデータセットから除外する
- また、意味のないレビューもデータセットから除外する



# データ成形

- 抽象化の例

raw source code

```
public PageProperties getProperties() {  
    if (hasProperties()) {  
        return properties;  
    } else {  
        return null;  
    }  
}
```

abstracted code

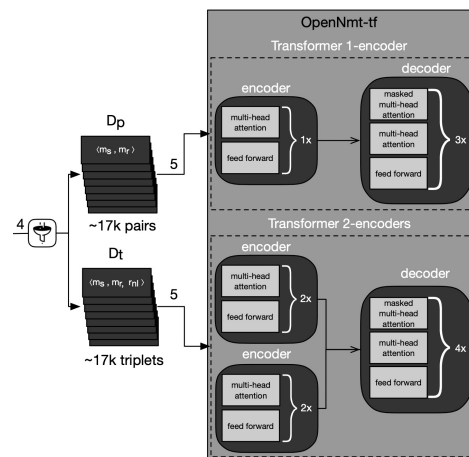
```
public TYPE_1 METHOD_1 ( ) { if ( METHOD_2 ( ) )  
{ return properties ; } else { return null ; } }
```

Fig. 2: Example of abstraction.

# データ成形

- msとmrが同じであるトリプレット
  - msもしくはmrの文字数が100字以上であるトリプレット
  - レビューが複数あるトリプレット
  - 重複しているトリプレット
- を取り除き，最終的に残った17194のトリプレットをデータセットとする
  - このデータセットをDtとし，Dtからレビューを除き，msとmrのペアからなるデータセットをDpとする

データセットは，トレーニングセット（80％），評価セット（10％），テストセット（10％）に分割している



# 機械学習

- msの入力からmrを生成する1-encoderモデルと、msとレビューの入力からmrを生成する2-encoderモデルを学習する
- ハイパーパラメータの探索にはTree Parzen Estimator [27][28]を使用し、50000ステップの学習を行った
- ハイパーパラメータの決定後はBeam Searchを用いて、複数個の出力を得る

[27] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 2546–2554. [Online]. Available: <http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf>

[28] J. Bergstra, D. Yamins, and D. Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures,” in *International conference on machine learning*, 2013, pp. 115–123.

# モデルの評価

- 1-encoderモデルと， 2-encoderモデルに対し， テストセットを用いて評価を行う
- msに対し， 予測されたコードとmrが同一である場合， 予測は正しいと考える．
- 自動的に生成されたテキストの品質を評価するために使用される指標であるBLEUスコア[36]を計算する
- 予測されたコードを参照コードに変換するために必要な， トークンの編集（挿入， 削除， 置換）の最小数を表すLevenshtein距離[37]を計算する

[36] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: A method for automatic evaluation of machine translation,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL ’02, 2002, pp. 311–318.

[37] V. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions and Reversals,” *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

# モデルの評価

- msとmrのみでは正しく予測された割合は**16%**であったのに対し、レビューを含むと**31%**に上昇した

| Beam Size        | Perfect Predictions |        | BLEU-4 |        |          | Levenshtein distance |        |          |
|------------------|---------------------|--------|--------|--------|----------|----------------------|--------|----------|
|                  | #                   | %      | mean   | median | st. dev. | mean                 | median | st. dev. |
| <b>1-encoder</b> |                     |        |        |        |          |                      |        |          |
| <b>1</b>         | 50                  | 2.91%  | 0.7706 | 0.8315 | 0.1929   | 0.2383               | 0.2000 | 0.1670   |
| <b>3</b>         | 156                 | 9.07%  | 0.8468 | 0.8860 | 0.1419   | 0.1726               | 0.1454 | 0.1427   |
| <b>5</b>         | 200                 | 11.63% | 0.8644 | 0.8980 | 0.1317   | 0.1554               | 0.1271 | 0.1348   |
| <b>10</b>        | 271                 | 15.76% | 0.8855 | 0.9145 | 0.1166   | 0.1355               | 0.1092 | 0.1247   |
| <b>2-encoder</b> |                     |        |        |        |          |                      |        |          |
| <b>1</b>         | 209                 | 12.16% | 0.8164 | 0.8725 | 0.1863   | 0.1849               | 0.1422 | 0.1734   |
| <b>3</b>         | 357                 | 20.77% | 0.8762 | 0.9244 | 0.1484   | 0.1321               | 0.0838 | 0.1468   |
| <b>5</b>         | 422                 | 24.55% | 0.8921 | 0.9376 | 0.1351   | 0.1173               | 0.0696 | 0.1366   |
| <b>10</b>        | 528                 | 30.72% | 0.9142 | 0.9543 | 0.1169   | 0.0953               | 0.0519 | 0.1204   |

TABLE II: Quantitative results: Perfect predictions, BLEU-4, and Levenshtein distance achieved by the models

# まとめ

- コードレビュープロセスの一部を自動化するモデルの提案
- レビュー用に提出されたコード，レビュー，修正されたコードを成形して機械学習を行なった
- msとmrのみでは正しく予測された割合は**16%**であったのに対し， msとmrとレビューの**3種類**だと**31%**であった

# 所感

- 正しく予測できた割合の低さから、予測モデルを作成し精度を高めることの難しさを感じた
- なぜデータ収集の際にJavaのプロジェクトを選んだのかがわからなかった