

AutoTransform: Automated Code Transformation to Support Modern Code Review Process

Patanamon Thongtanunam,
Chanathip Pornprasit, Chakkrit Tantithamthavorn

44th International Conference on Software Engineering (ICSE2022)

発表者：松尾 和輝

背景

- コードレビュー
 - メインリポジトリに統合する前に、新しく開発されたコードの品質が標準を満たしていることを確認する作業^[11, 37]
 - レビューアーはコードを統合するのに十分な品質があることを確認する必要がある
 - レビューアーはレビューを行うのに多大な労力が必要
 - 時間的, 人的制限
 - 大量のレビューをミスなく行うことは困難

- レビューを支援する研究が行われている
- 自動プログラム修正, トリアージ, 優先順位付け

[11] Alberto Bacchelli and Christian Bird. 2013. Expectations, Outcomes, and Challenges of Modern Code Review. In Proceedings of ICSE. 712-721.

[37] Laura MacLeod, Michaela Greiler, Margaret-Anne Storey, Christian Bird, and Jacek Czerwinka. 2017. Code Reviewing in the Trenches: Challenges and Best Practices. IEEE Software 35, 4 (2017), 34-42.

先行研究

- コードをレビューされ承認されたバージョンに自動的に変換するNMTモデルを提案
 - アプローチ

コードの抽象化

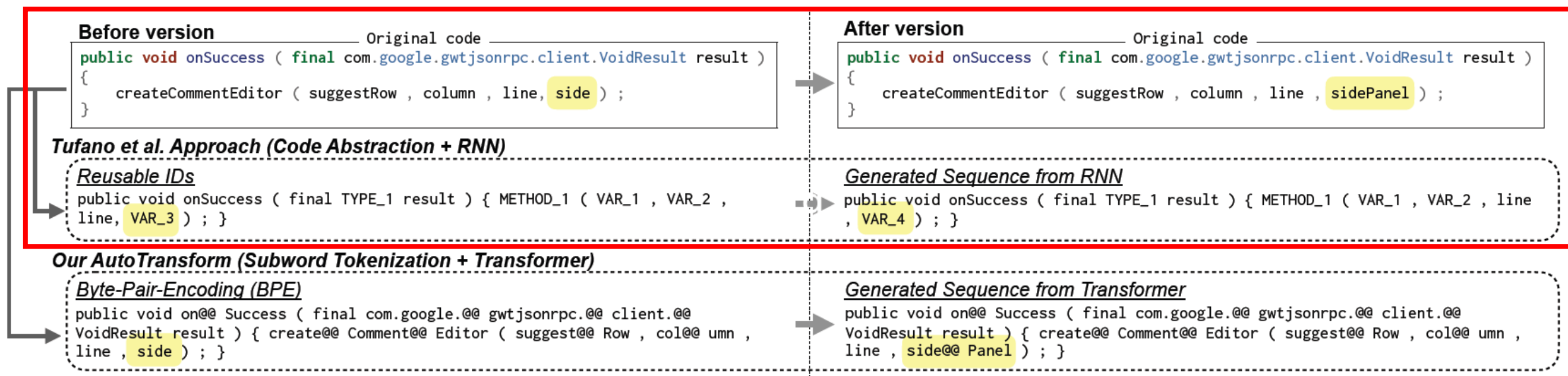
side => VAR_3
google.gwt => TYPE_1
create => METHOD_1

NMTの構築

変更前のコードから、
変更後のコードを予測
するRNNモデルの学習

予測の生成

変更前のコードを入力し、
変更後のコードを
予測



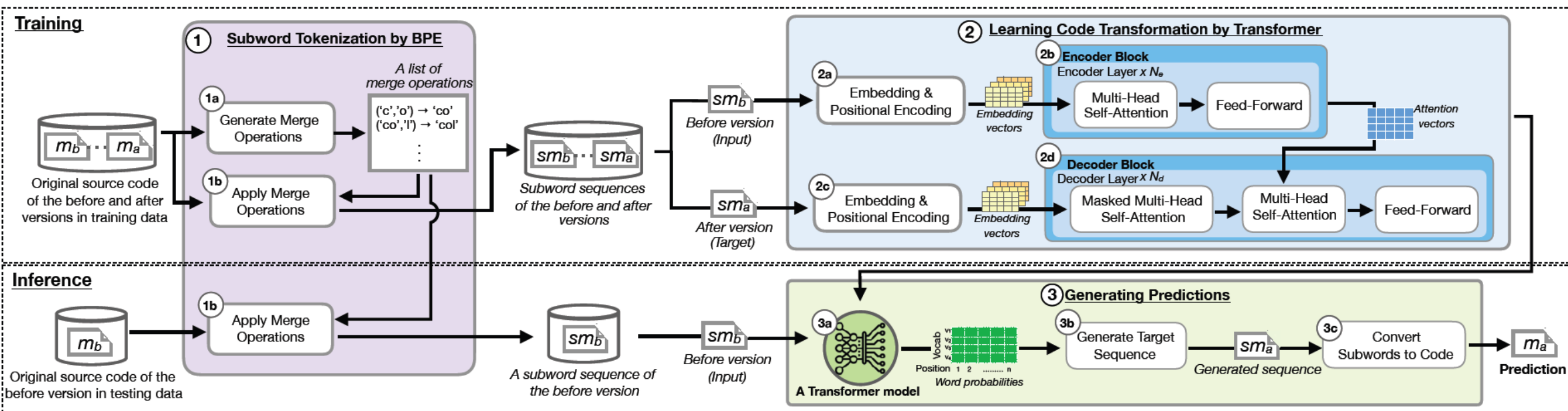
先行研究 | 問題点

- 問題点1 | 変更後コードに新たなトークンがあると予測困難
 - 変更前コードトークンのsideから変更後コードトークンのsidePanelを予測できない
- 問題点2 | 入力文字列が長いと予測困難
 - RNNは入力初期のトークンを忘れてしまう

本研究の目的
以上の問題を解決するモデルAutoTransformの提案

提案手法 | 概要

- Step
 - ①BPEを用いたサブワード列の生成
 - ②Transformerを用いた学習
 - ③学習モデルを用いた予測の生成



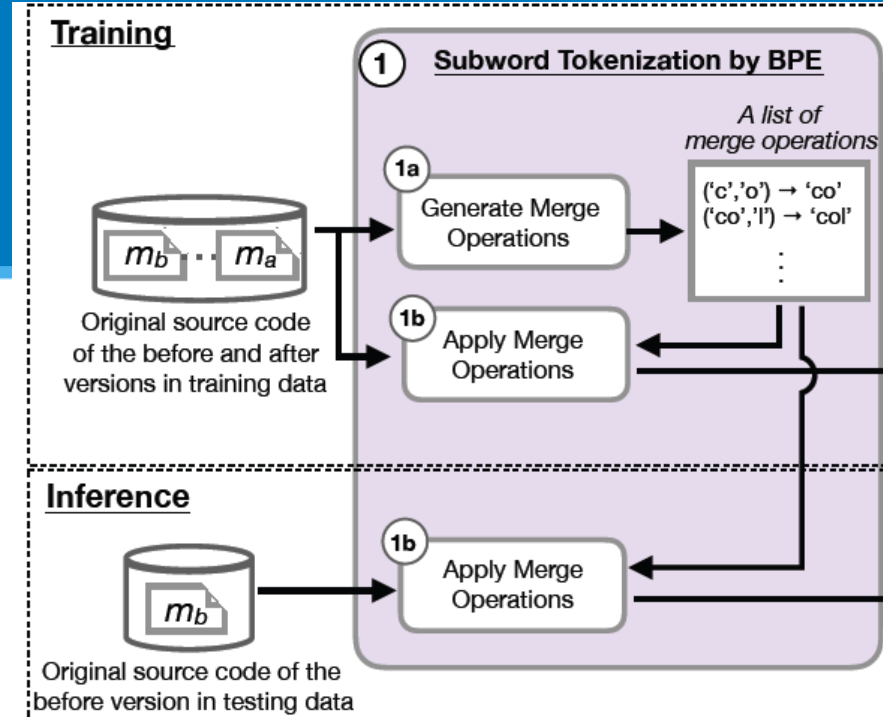
Step① | BPE

- 文字列の分割方法

- 与えられたデータを1文字単位で分割
- 例) 'low', 'lowest', 'newer', 'wider'
=> 'l o w', 'l o w e s t', 'n e w e r', 'w i d e r'

- 指定された回数だけ頻出する文字を結合

- 例)
'r' + '</w>' -> 'r</w>'
'l' + 'o' -> 'lo'
'lo' + 'w' -> 'low'
'e' + 'r</w>' -> 'er</w>'
</w>: end of word



Step① | サブワード列の生成

- 入力されたコードにBPEを適用

入力されたコード

Original code

```
public void onSuccess ( final com.google.gwtjsonrpc.client.VoidResult result )  
{  
    createCommentEditor ( suggestRow , column , line, side ) ;  
}
```



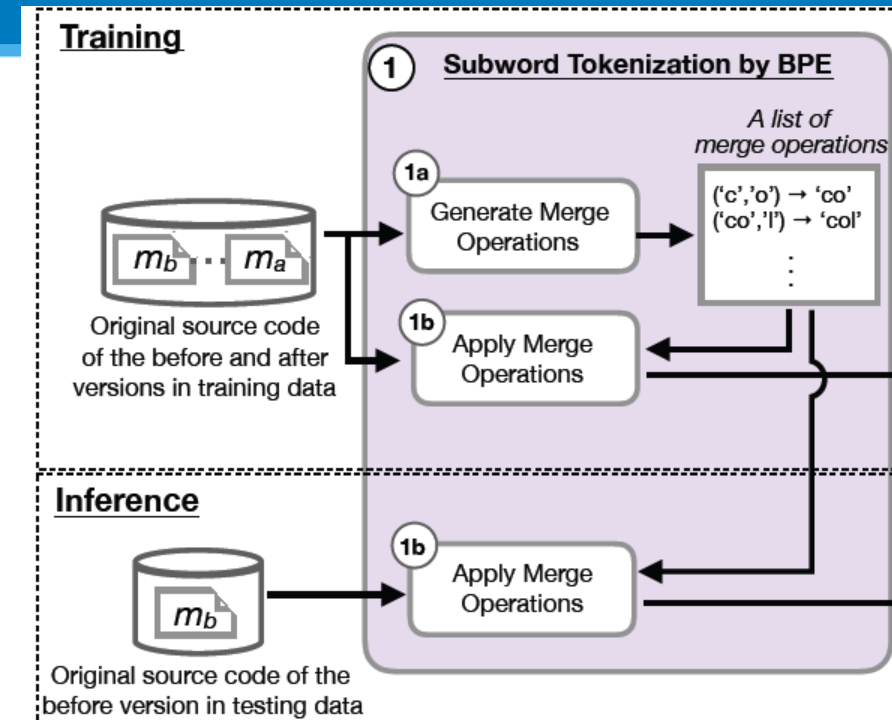
サブワード列

Byte-Pair-Encoding (BPE)

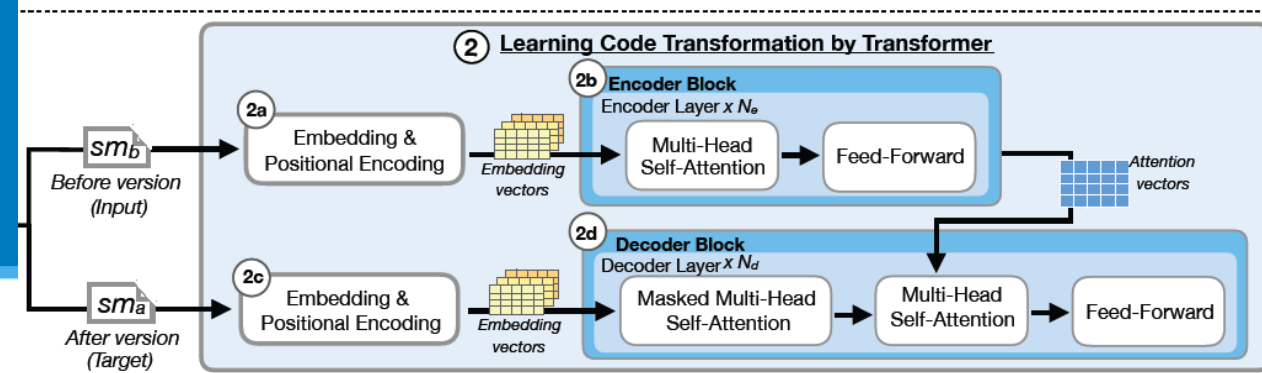
```
public void on@@ Success ( final com.google.@@ gwtjsonrpc.@@ client.@@  
VoidResult result ) { create@@ Comment@@ Editor ( suggest@@ Row , col@@ umn ,  
line , side ) ; }
```

学習データに”side@@”と”Panel”があれば, “sidePanel”を生成可能

- 変更前, 変更後コードから, それぞれサブワード列を生成する

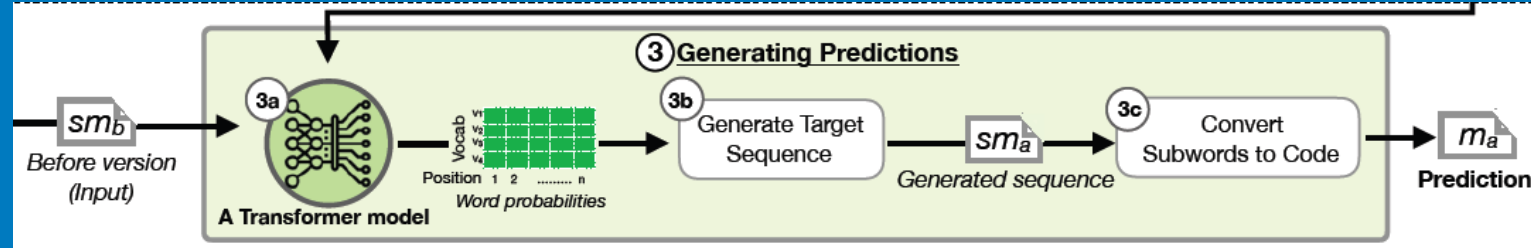


Step② | 学習



- Transformerを用いて, 変更前コードから変更後コードを予測するモデルを学習
 - Encoder部分に変更前コード
 - Decoder部分に変更後コード
 - をそれぞれ入力
 - Transformerを用いて学習

Step③ | 予測



- 学習したモデルを用いて， 変更前コードから変更後コードを予測
 - ビームサーチ法^[53]を利用して変更後コードを生成
 - ビーム幅が3ならば， 変更後コードを3パターン生成する
- サブワードの統合
 - @@で終わるサブワードとそれに続くサブワードを統合
 - col@@とumnというサブワードはcolumnに変換

[53] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In NIPS. 3104-3112.

リサーチクエスチョン

- RQ1

- AutoTransformは先行研究のアプローチよりも優れたコード変換が可能か？

- RQ2

- AutoTransformの構成要素はどのような貢献をしているか？

データセット

ー 先行研究で扱われたデータを利用

- メソッドレベルのコードを収集

Dataset	Method size	w/o New Tokens	w/ New Tokens
Android	Small	4,437 (18%)	20,640 (82%)
	Medium	4,593 (16%)	24,547 (84%)
Google	Small	2,289 (20%)	9,070 (80%)
	Medium	2,833 (20%)	11,624 (80%)
Ovirt	Small	4,734 (17%)	23,282 (83%)
	Medium	6,229 (16%)	33,275 (84%)

- メソッドサイズ

- Small : トークン数が50以下のメソッド
- Medium : トークン数が50~100のメソッド

- 変更タイプ

- w/o : 変更後コードに新たなトークンが含まれていないメソッド
- w : 変更後コードに新たなトークンが含まれているメソッド

ー $3 \times 2 \times 2 = 12$ のデータセットでそれぞれ学習

結果 | RQ1 提案手法の有効性

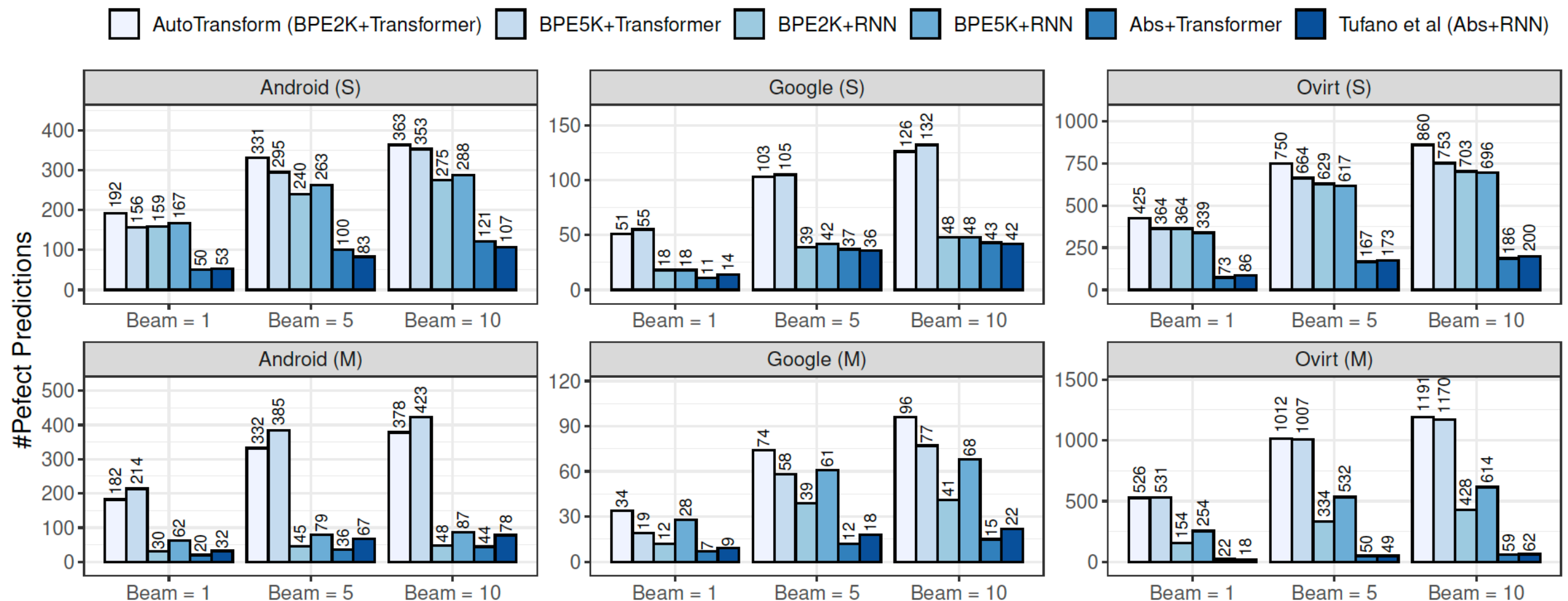
- 既存手法では予測できないコードも予測可能
- 入力文字列が長くても予測可能
- #PP

完全予測：生成されたコードと正解のコードが完全一致しているか判定

			Beam width = 1		Beam width = 5		Beam width = 10	
Dataset (Method Size)	Change Type	#Test	AUTOTRANSFORM #PP	Tufano <i>et al.</i> #PP	AUTOTRANSFORM #PP	Tufano <i>et al.</i> #PP	AUTOTRANSFORM #PP	Tufano <i>et al.</i> #PP
Android (Small)	w/o new tokens	443	84	53	125	83	130	107
	w/ new tokens	2,064	108	0	206	0	233	0
Google (Small)	w/o new tokens	228	11	14	22	36	29	42
	w/ new tokens	907	40	0	81	0	97	0
Ovirt (Small)	w/o new tokens	473	73	86	132	173	145	200
	w/ new tokens	2,328	352	0	618	0	715	0
Android (Medium)	w/o new tokens	459	58	32	85	67	89	78
	w/ new tokens	2,454	124	0	247	0	289	0
Google (Medium)	w/o new tokens	283	16	9	28	18	33	22
	w/ new tokens	1,162	18	0	46	0	63	0
Ovirt (Medium)	w/o new tokens	622	111	18	179	49	199	62
	w/ new tokens	3,327	415	0	833	0	992	0
Total	w/o new tokens	2,508	353	212	571	426	625	511
	w/ new tokens	12,242	1,060	0	2,031	0	2,389	0
	Both	14,750	1,413 (+567%)	212	2,602 (+511%)	426	3,014 (+490%)	511

結果 | RQ2 構成要素の貢献

- AutoTransformの構成要素を変化させて評価
 - コード抽象化 => BPEにより, 最低284%精度が向上
 - RNN => Transformerにより, 最低17%精度が向上



まとめ

- 目的
 - 新しいトークンや，長い文字列を扱えるようにした AutoTransform の提案
- アプローチ
 - 新しいトークンについては BPE，長い文字列については Transformer を活用し，学習を行う
- 結果
 - 先行研究では扱えなかった新しいトークンや，長い文字列も，提案手法では生成可能であることを示した

所感

- 変更後コードに新たなトークンがあると予測困難ってどういうこと？？？
 - 新たなトークンがない変更って存在するの？
- 完全一致でこれほど精度が高いのが気になる
 - ほんの少ししか変更していないものが多いのか？
- なんでBPEを用いると精度が上がるのかがいまいちわからない
 - 新たなトークンがあるケースが底をあげているだけか