

Discovering Repetitive Code Changes in Python ML Systems

Malinda Dilhara
malinda.malwala@colorado.edu
University of Colorado Boulder
USA

Nikhith Sannidhi
nikhith.sannidhi@colorado.edu
University of Colorado Boulder
USA

Ameya Ketkar*
ketkara@uber.com
Uber Technologies Inc.
USA

Danny Dig
danny.dig@colorado.edu
University of Colorado Boulder
USA

44TH International Conference On Software Engineering (ICSE 2022)

発表者：井町 瑠貴

背景 | ソフトウェア変更の繰り返し

長年、ソフトウェア変更パターンを利用し、多くのソフトウェア進化タスクなどを自動化

- しかし、これらのほとんどがpythonやML系には対応していない

MLのためのツールを発展させるためには、MLシステムがどのように進化・維持するかを理解する必要がある

- 細かなコードの変更の繰り返しについて理解が不足

背景 | 動機となる例

CPatMiner(既存研究) [58]

- Java のコード変更パターンをマイニング
- 重複するリファクタリングを考慮していない

R-CPatMiner(提案手法)

- Javaのリファクタリング推論ツールRminer^[85]をPythonに対応させ, CPatMinerと統合(JavaFyPy)

Listing 2: Specifies the device (CPU) for operations executed in the context and move method `_init_model` to parent class

```
class _FERNeuralNet():
+   def _init_model(self):
+       with tf.device('/cpu:0'):
+           B, H, T, _ = q.get_shape().as_list()
...
class TimeDelayNN(_FERNeuralNet):
-   def _init_model(self):
-       B, H, T, _ = q.get_shape().as_list()
```

[58] H. Nguyen, T. N. Nguyen, D. Dig, S. Nguyen, H. Tran, and M. Hilton. 2019. GraphBased Mining of In-the-Wild, Fine-Grained, Semantic Code Change Patterns. In ICSE 2019. IEEE Computer Society, Los Alamitos, CA, USA, 819–830. <https://doi.org/10.1109/ICSE.2019.00089>

[85] N. Tsantalis, A. Ketkar, and D. Dig. 5555. RefactoringMiner 2.0. TSE 2020 01 (jul 5555), 1–1. <https://doi.org/10.1109/TSE.2020.3007722>

提案手法の概要

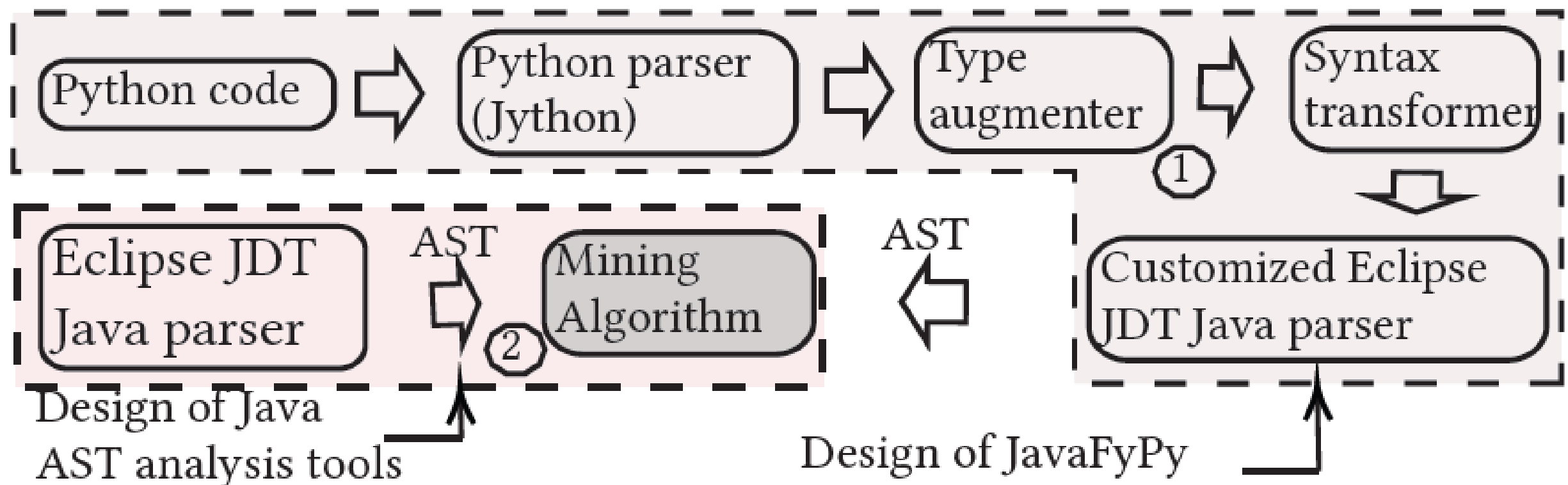


Figure 1: Design of JAVAfYPY and existing AST analysis tools

リサーチクエスチョン

RQ1 : MLで頻出のコード変更のパターンは何か
自動化を必要としているパターンは何か

- R-CPatMiner を用いてマイニング
- 主題分析を適用

RQ2 : リファクタリングに注目することで
パターンマイニングがどのように改善されるか

RQ3 : R-CPatMiner, PyCPatMiner, PyRMinerの
実行性能はどの程度か

- 実行時間を比較

対象システム

既存研究^[22]で用いられたコーパスを利用

- スター数が50以上
- アーカイブされていないpythonリポジトリ
- 以下の条件のものを除外
 - データ収集日の6ヶ月以内に活動がないもの
 - 貢献者が3人未満のリポジトリ
- ソースコード：5800万行， ファイル：15万

[22] Malinda Dilhara, Ameya Ketkar, and Danny Dig. 2021. Understanding Software2.0: A Study of Machine Learning Library Usage and Evolution. ACM Trans. Softw. Eng. Methodol. 30, 4, Article 55 (July 2021), 42 pages. <https://doi.org/10.1145/3453478>

RQ1 | 検出されるパターンの特徴

R-CPatMiner コーパスに対して実行

- 349406 個の変更グラフ
- 28308個のコード変更パターン

開発者の53%がパターンを100%,
79%が50%, 91%が10%を共有している

プロジェクトの36%がパターンを100%,
60%が50%, 91%が10%のパターンを共有している

開発者やプロジェクトの間に浸透している
パターンを抽出していることがわかる

RQ1 | パターンの傾向

| 以下の4つの傾向が見られた

- コンテキストマネージャへの変換
- forループの変換
- API の使用方法の更新
- 高度な言語機能の使用

| これらの変更は手動で行われており, 71%の開発者がIDEに自動化を求めている

RQ2 | リファクタリングによる影響

R-CPatMinerはPyCPatMinerよりも16%多く変更されたメソッドを処理

- リファクタリングを考慮することは有効

Table 3: Analysed data set and execution time.

	Java CPATMiner	Py- CPATMiner	R-CPATMiner
Total changed methods	824K	1M	1.16M
Total AST nodes of changed methods	92M	4.5B	4.6B
Total changed graph nodes	8M	4M	4.7M
Total patterns	17K	24K	28K
Execution time	<8hours	<12hours	<19hours

RQ3 | 結果

実行時間

- R-CPatMiner : 19時間未満
- PyCPatMiner : 12時間未満
- Java-CPatMiner : 8時間未満

JavaとPythonのコーパスは多様であり、
公平な比較は難しい

- CPatMinerの主要な目的は、パターンのデータベースを構築するために毎週コーパスをマイニングすることである
- 目標を達成するためには妥当な実行時間である

結論

- | PythonベースのMLソフトウェアシステムにおける、きめ細かいコード変更パターンに関する最初の調査を行った
- | 22の変更パターンを4つに分けて提示し、そのうち10はMLに特化したものであった
- | アンケートでは、71%の回答者がこれらのパターンをIDEで自動化することを希望している

所感

- | 最近，既存の技術・ツールをML開発に応用する研究を見かけるようになったと感じる
- | 性能比較で実行時間を比較していたが，必要であったのか疑問を感じた

補足資料

補足 | パターンの分類

Table 1: Triangulating source code mining results with survey responses: 4 major trends, the pattern groups for each trend, and whether that pattern is specific to ML code (column ML). Column I shows the number of instances for each pattern. Column R shows number of survey respondents. Next columns indicate their responses to survey Q2 (How often these changes happen in ML code?) and Q3 (How often have you manually performed this change?), with frequency: Very Often (VO), Often (O), Rare (R), and Never (N). Q4 (Would you like to have this change automated by a tool?) response: Yes, No, Already Automated (AA).

Static Analysis					Survey Responses											
					Q ₂				Q ₃				Q ₄			
Trend	P	Pattern	ML ¹	I	R	VO	O	R	N	VO	O	R	N	Yes	No	AA
Move to with statement and use Context managers	P1	Read, write, traverse data	×	467	6	67%	17%	-	17%	83%	-	17%	-	100%	-	-
	P2	Disable or enable gradient calculation	✓	92	5	100%	-	-	-	80%	-	20%	-	100%	-	-
	P3	Swap ML training devices	✓	21	3	100%	-	-	-	67%	33%	-	-	67%	33%	-
	P4	Change name and variable scopes in DL networks	✓	106	7	57%	43%	-	-	57%	29%	14%	-	57%	43%	-
	P5	Execute dependencies of a Tensorflow graphs	✓	57	3	100%	-	-	-	100%	-	-	-	33%	67%	-
	P6	Temporarily change configurations of libraries	×	59	2	-	50%	50%	-	100%	-	-	-	50%	50%	-
	P7	Transform to context managers in pytest	×	335	6	33%	50%	17%	-	67%	33%	-	-	67%	33%	-
	P8	Use context managers to open temporary directories	×	100	6	33%	50%	-	17%	67%	-	33%	-	83%	17%	-
Dissolve for loops, into domain specific abstractions	P9	Transform to optimized operations in NumPy	✓	179	10	100%	-	-	-	90%	-	10%	-	100%	-	-
	P10	Transform to operations in <i>List</i> or <i>Dictionary</i>	×	24	3	67%	33%	-	-	33%	-	67%	-	33%	67%	-
	P11	Transforming to Python built in functions	×	15	2	-	100%	-	-	-	-	100%	-	100%	-	-
	P12	Transform functions in <i>String</i>	×	14	2	-	50%	50%	-	100%	-	-	-	100%	-	-
	P13	Transform to set operations	×	7	2	100%	-	-	-	50%	-	50%	-	100%	-	-
Update API usage, (212)	P14	Migrating to APIs ML libraries	✓	26	5	100%	-	-	-	100%	-	-	-	100%	-	-
	P15	Transform Matrix	✓	82	6	83%	17%	-	-	100%	-	-	-	33%	50%	-
	P16	Change data visualization	✓	28	2	50%	50%	-	-	50%	-	50%	-	50%	50%	-
	P17	Composite ML APIs	✓	22	5	100%	-	-	-	100%	-	-	-	100%	-	-
	P18	Update Container	×	115	5	60%	-	40%	-	60%	-	40%	-	80%	20%	-
	P19	Update Type of Matrices	✓	16	4	100%	-	-	-	50%	-	50%	-	50%	50%	-
Use advanced Language Features	P20	Simplify conditional statement	×	24	3	-	33%	67%	-	67%	-	-	33%	33%	-	67%
	P21	Migrate from Dict, Set, List constructors to literals	×	42	4	-	25%	75%	-	25%	-	-	75%	25%	-	75%
	P22	Transform to Python List, Dict, or Set Comprehension	×	349	6	-	33%	67%	-	33%	-	-	67%	33%	-	67%

¹ ML specific Patterns: i.e., patterns related to ML techniques. More than 80% developers confirmed they happen in ML very often or often