

SOAR: A Synthesis Approach for Data Science API Refactoring

Ansong Ni*
Yale University
New Haven, USA
ansong.ni@yale.edu

Daniel Ramos*
INESC-ID/IST, U. Lisboa, Portugal
Carnegie Mellon University, USA
danielrr@cmu.edu

Aidan Yang
Queen's University
Kingston, Canada
a.yang@queensu.ca

Inês Lynce
INESC-ID/IST, U. Lisboa
Lisboa, Portugal
ines.lynce@tecnico.ulisboa.pt

Vasco Manquinho
INESC-ID/IST, U. Lisboa
Lisboa, Portugal
vasco.manquinho@tecnico.ulisboa.pt

Ruben Martins
School of Computer Science
Carnegie Mellon University
Pittsburgh, USA
rubenm@cs.cmu.edu

Claire Le Goues
School of Computer Science
Carnegie Mellon University
Pittsburgh, USA
clegoues@cs.cmu.edu

A. Ni *et al.*, "SOAR: A Synthesis Approach for Data Science API Refactoring," *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 112-124

背景

- データサイエンス系APIの進化が日々進んでいる
- APIが更新される毎にAPIを使用しているソフトウェアを修正する必要がある
- API更新に伴う修正は欠陥混入の危険性や手間の増加を招く

→API更新があった場合に自動修正があると便利

既存研究と問題点

- 既存研究

- 開発者がリファクタリング（API移行）したプログラムを学習して自動化する手法[49]
- 異なる言語間でもリファクタリングの自動化手法が構築されている[51]

- 問題点

- 新たなライブラリやバージョンでは使えないため学習を要する手法は理想的でない

[49] M. Fazzini, Q. Xin, and A. Orso, “APIMigrator: An API-Usage Migration Tool for Android Apps,” in *Proc. International Conference on Software Engineering*, IEEE / ACM, 2020.

[51] H. Zhong, S. Thummalapenta, T. Xie, L. Zhang, and Q. Wang, “Mining API mapping for language migration,” in *Proc. International Conference on Software Engineering*, pp. 195–204, ACM, 2010.

目的とアプローチ

- 目的

- 学習データを要しない自動API移行手法SOARを構築する

- アプローチ

- 自然言語処理とプログラム合成を組み合わせる
- 合成したコードのエラーメッセージを利用してエラーに対処する

APIのリファクタリング

- API移行

- あるAPIを用いている修正済みのソースコードから別のAPIを用いている未修正のソースコードへ移す
- 移行時にそれぞれのAPIのドキュメントを熟読する必要がある
- 対応するAPI, 引数の型, 引数に入れる値など間違えたものを使用すると欠陥混入の原因になる

API移行の実例

- 返り値, 引数の型や名前, 引数の数を整える (A,B,C)
- 引数の順番を変える (A,C)
- 対応するメソッドが複数 (C)

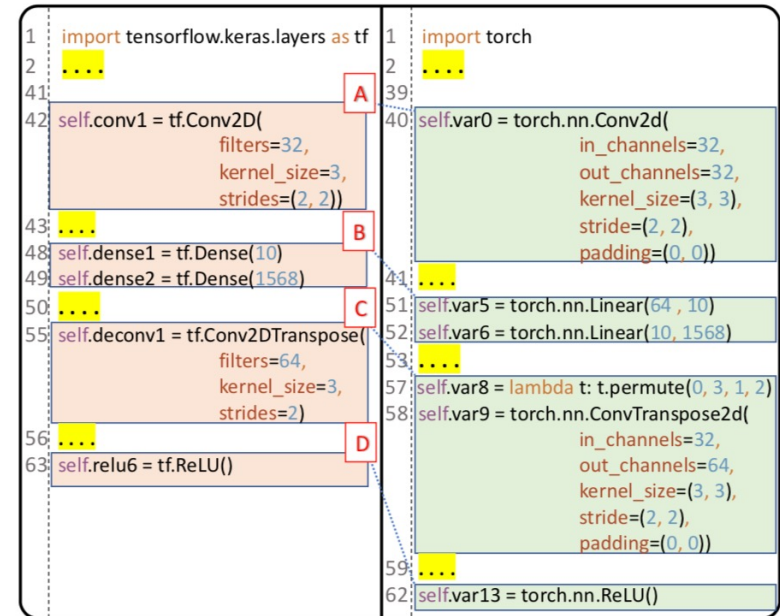


Fig. 1: An example of how SOAR refactors a program written with TensorFlow (left) to using PyTorch (right). Note that the whole program consists of 15 APIs calls to TensorFlow, though we only show four blocks of them (*i.e.*, A, B, C and D) for brevity. SOAR can migrate the full program in 161 seconds.

TensorFlow → PyTorch

SOARの概要

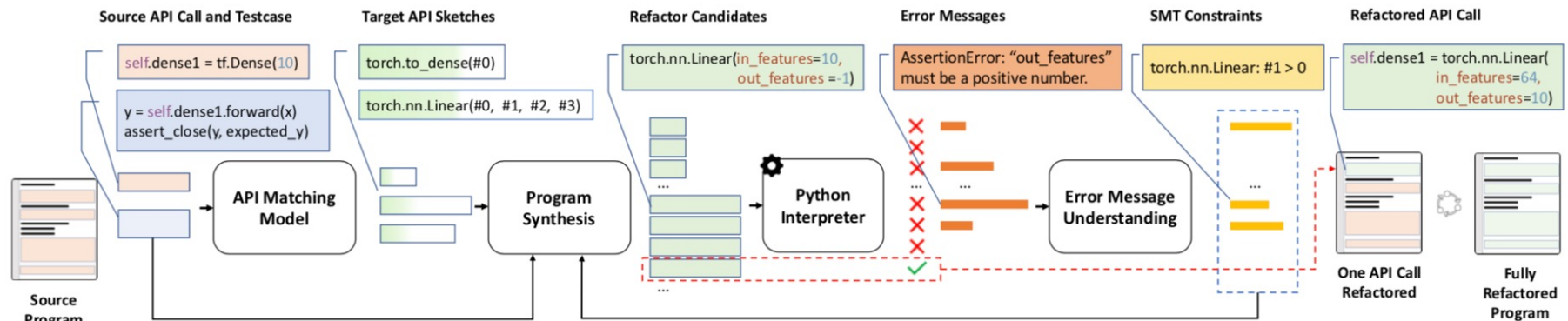


Fig. 2: Overview of SOAR's architecture.

- 対応APIドキュメント特定
- プログラム合成
- エラーメッセージ理解

対応APIドキュメント特定

- 移行元APIと移行先APIのドキュメントから詳細な説明が書かれた箇所を特定
- 類似する文書を特定するために以下のアルゴリズムを使用
 - TF-IDF[25]
 - 代表的な単語を含む文書を特定することで類似する文書を探す
 - 同じ意味でも使用する単語や文章が異なれば類似文書として特定できない
 - Tfidf-GloVe[26]
 - 大きな自然言語コーパスで学習したGloVe埋め込みを使用したTF-IDF

[25] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information processing & management, vol. 24, no. 5, pp. 513–523, 1988.

[26] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in Proc. Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543, ACL, 2014.

プログラム合成

- Program Sketching

- 移行先のAPIで引数が不明な部分を穴あき状態にして作成
- 例)

```
tf.keras.layers.Conv2D  
(filters=32, kernel_size=3, strides=(2, 2))
```

↓

```
torch.nn.  
Conv2d (#1,#2,(#3,#4),stride=(#5,#6),padding=(#7,#8))
```

プログラム合成

- Program Enumeration

- 穴あき部分の穴を埋めるプログラムの候補を列挙
- SMTソルバを用いて穴埋めの候補を抽出
 - 各穴には正確に1つの引数が含まれている
 - 各穴には正しい型の引数のみが含まれている

- Specification Constraints

- APIドキュメントからSMTソルバの制約となる情報を抽出
- 制約を増やすことで検索スペースを節約

エラーメッセージ理解

- step1

- 下位概念の抽出

- step2

- エラー発生原因のパラメータ引数と制約を特定

- step3

- 変更内容反映

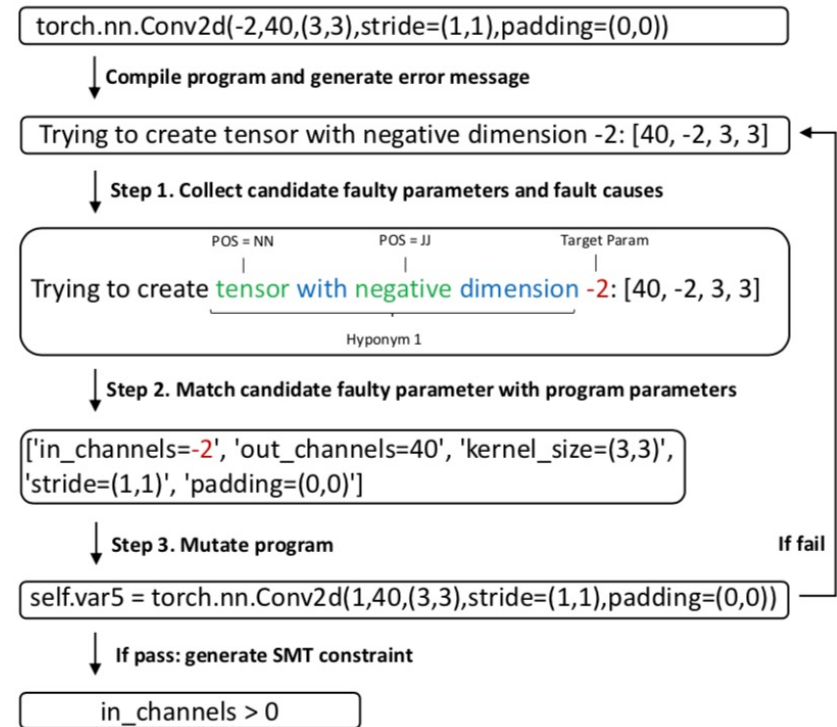


Fig. 5: Example error message to SMT constraint pipeline using hyponym 1.

評価実験

- データセット
 - TensorFlow→PyTorchのために20件のTensorFlowチュートリアルプログラムを収集
 - dplyr(R)→pandas(Python)のために20件のkaggleチュートリアルプログラムを収集
- リサーチクエスチョン
 1. SOARのAPI移行はどの程度効果的か
 2. SOARの各構成要素がパフォーマンスに与える影響はどれほどか
 3. ディープラーニングライブラリの移行以外のドメインに一般化できるか

SOARのAPI移行精度

- 80%のプログラムでAPI移行が成功した
- SOARの一部の機能を使わなかった場合、性能が大きく劣る

TABLE II: Execution time for the deep learning library migration task in each of the 20 benchmarks.

	SOAR	SOAR w/o Specs.	SOAR w/o Err. Msg.
conv_pool_softmax(4L)	1.60	23.02	14.35
img_classifier(8L)	12.82	336.00	65.66
three_linear(3L)	3.18	2.34	21.07
embed_conv1d_linear(5L)	5.27	123.85	16.90
word_autoencoder(3L)	1.81	1.46	2.64
gan_discriminator(8L)	12.80	timeout	252.20
two_conv(4L)	16.69	timeout	15.09
img_autoencoder(11L)	160.97	391.09	487.54
alexnet(20L)	425.22	timeout	66.13
gan_generator(9L)	412.47	timeout	timeout
lenet(13L)	280.91	timeout	timeout
tutorial(10L)	6.04	timeout	58.29
conv_for_text(11L)	9.04	timeout	32.29
vgg11(28L)	40.83	timeout	132.67
vgg16(38L)	82.05	timeout	139.27
vgg19(44L)	83.99	timeout	189.90
densenet_main1(5L)	timeout	timeout	timeout
densenet_main2(3L)	timeout	timeout	timeout
densenet_conv_block(6L)	timeout	timeout	timeout
densenet_trans_block(3L)	timeout	timeout	timeout

SOARの各構成要素が与える影響

- 通常のTF-IDFの方が性能が高いことがわかる
- APIメソッド名がほぼ同じ場合はTF-IDFで十分であると考えられる

TABLE III: Execution time and average API ranking for each of the 20 benchmarks using TF-IDF and GloVe models.

	SOAR w/ TF-IDF		SOAR w/ TfIdf-GloVe	
	Time(s)	Avg. Ranking	Time(s)	Avg. Ranking
conv_pool_softmax(4L)	1.60	1.0	1.56	1.0
img_classifier(8L)	12.82	2.8	31.04	2.8
three_linear(3L)	3.18	8.0	7.70	8.0
embed_conv1d_linear(5L)	5.27	2.4	7.75	2.4
word_autoencoder(3L)	1.81	1.0	1.52	1.0
gan_discriminator(8L)	12.80	3.5	37.01	2.8
two_conv(4L)	16.69	1.0	13.75	1.0
img_autoencoder(11L)	160.97	1.9	166.34	1.9
alexnet(20L)	425.22	2.0	428.42	2.0
gan_generator(9L)	412.47	2.1	1892.86	2.6
lenet(13L)	280.91	4.3	timeout	89.1
tutorial(10L)	6.04	2.4	21.31	2.4
conv_for_text(11L)	9.04	2.3	14.08	2.3
vgg11(28L)	40.83	1.8	73.92	1.8
vgg16(38L)	82.05	1.6	114.41	1.6
vgg19(44L)	83.99	1.5	114.98	1.5
densenet_main1(5L)	timeout	172.8	timeout	285.6
densenet_main2(3L)	timeout	10.0	timeout	285.5
densenet_conv_block(6L)	timeout	293.3	timeout	634.0
densenet_trans_block(3L)	timeout	291.0	timeout	662.7

他のドメインでの性能

- dplyr→pandasでは90%のプログラムで成功した(Fig. 6)
- dplyr→pandasではAPIメソッド名の類似度が低いため、GloVeの方が性能が高い(Fig. 7)

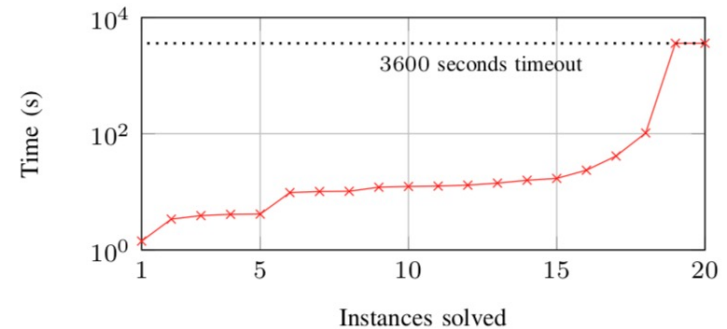


Fig. 6: Execution time for each benchmark of the dplyr-to-pandas task with a timeout of 3600 seconds.

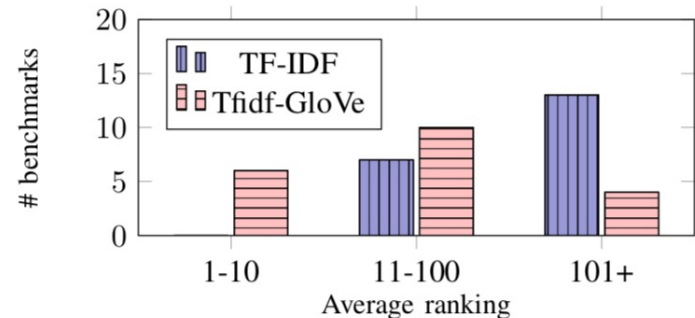


Fig. 7: Average ranking of the APIs for each of the 20 dplyr-to-pandas benchmarks.

まとめ

- データサイエンスドメインで学習データを用いない自動API移行手法SOARを構築
- 実験ではほとんどのAPIを自動的に移行できた
- SOARの構成要素であるプログラム合成, エラーメッセージ理解は性能改善に必要不可欠

所管

- 学習データを用いないという点で既存研究よりは気軽に使えそうだと思った
- **API**のソースコードの情報を加えることで精度，性能にどう影響するのか気になった
- 他のドメインでもどれほど通用するのか気になった