

研究発表

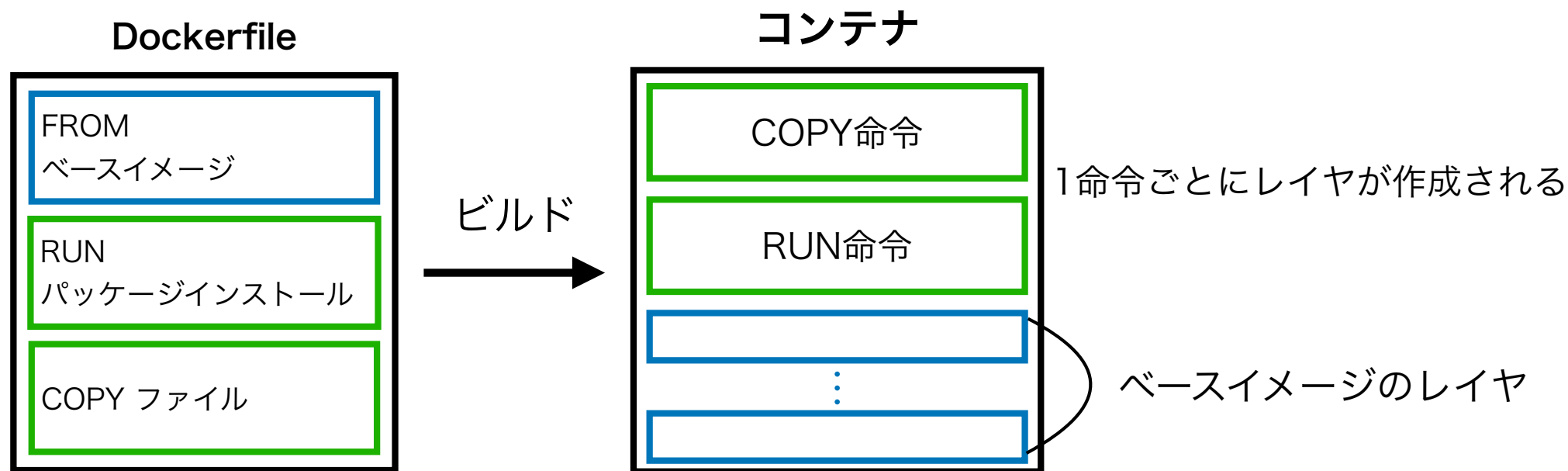
Dockerfileの修正履歴から抽出した パッチを用いた自動修正ツールの構築

辻 裕真

背景

Dockerコンテナ

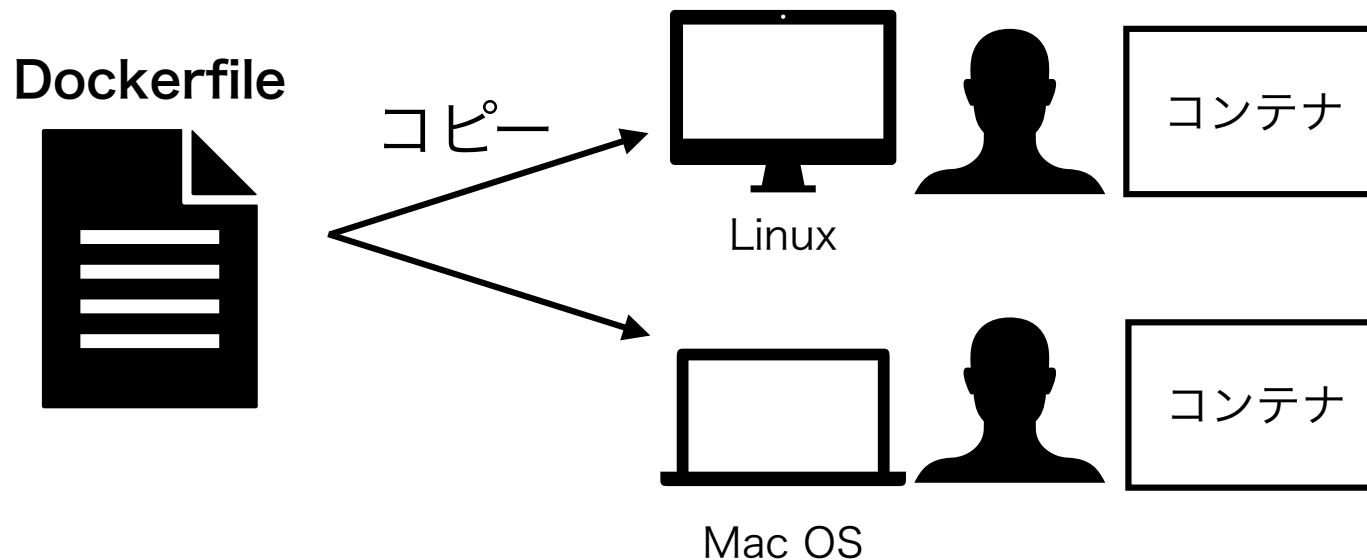
- ・ 少ないリソースで動作するアプリケーション実行環境
- ・ アプリケーション, ライブラリ, 設定ファイルなどが含まれる
- ・ Dockerfileをビルドすることでコンテナを作成できる



背景

Dockerコンテナ

- Dockerfileを共有することで容易にコンテナがコピーできる
- 異なる実行環境でも同一のアプリケーションを実行できる
- 近年のシステム開発においてDockerは頻繁に利用されている



背景

- Dockerfileはビルドの失敗率が高い(約34%)[1]
- Dockerfileのビルドエラー修正に要する時間は長い[2]
- 多くの利用者がいるコンテナの場合、影響が大きい



Dockerfileを自動修正することで開発者を支援する研究が行われている

[1]Jürgen Cito, Gerald Schermann, John Erik Wittern, Philipp Leitner, Sali Zumberi and Harald C. Gall. An Empirical Analysis of the Docker Container Ecosystem on GitHub. Proceedings of the IEEE/ACM 14th International Conference on Mining Software Repositories, pp. 323-333, 2017

[2]Yiwen Wu, Tao Wang, Yang Zhang and Huaimin Wang.

An Empirical Study of Build Failures in the Docker Context. Proceedings of the 17th International Conference on Mining Software Repositories, pp. 76-80, 2020

先行研究

- HenkelらはDockerfile自動修正ツールShipwrightを提案[3]
- 多数のDockerfileのビルド実行時のエラーログをクラスタリングし修正できそうなエラーを特定
- 人手で13種類のパッチを作成

パッチの例

log : “Unable to locate package(*)”
Dockerfile : “FROM ubuntu:(latest or 20.04)” → “FROM ubuntu:18.04”

- ビルドに失敗したDockerfileの約19%を修正できた

[3] Jordan Henkel, Denini Silva, Leopoldo Teixeira, Marcelo d'Amorim, Thomas Reps.
Shipwright: A Human-in-the-Loop System for Dockerfile Repair. Proceedings of the 43rd International Conference on Software Engineering, 2021

先行研究の課題

- ・ 環境の変化によりパッチが古くなってしまう
- ・ パッチの作成は手作業で行われているため限界がある



Dockerfileの修正履歴からパッチを自動で生成することで
より多くのビルドエラーを修正できるのではないか

目的とアプローチ

- ・ 目的

自動抽出したパッチを用いた自動修正ツールの構築

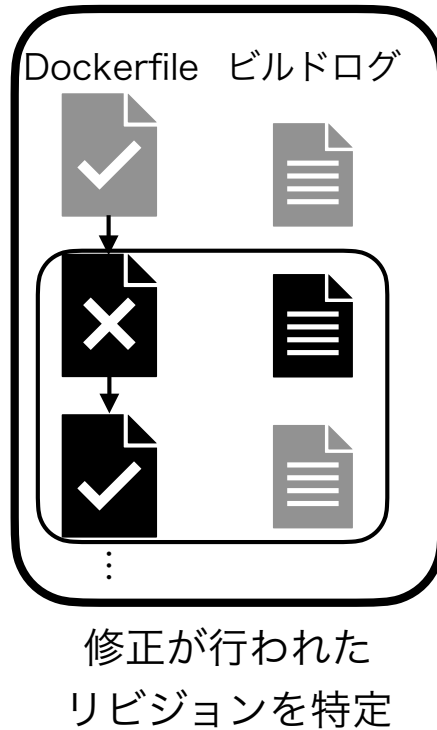
- ・ アプローチ

1. Dockerfileのビルド履歴と変更履歴から修正箇所を特定

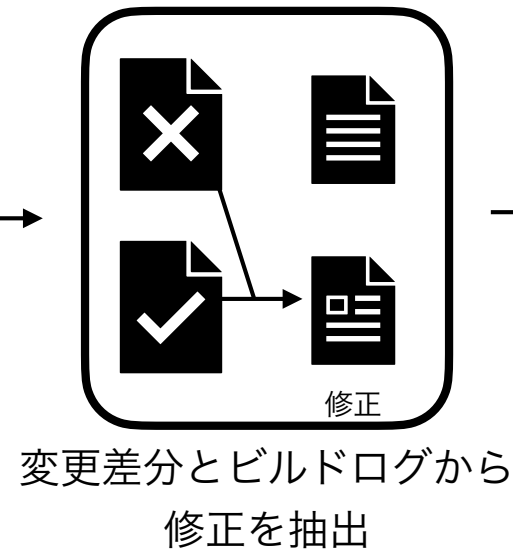
2. 機械学習を用いてパッチを作成して適用

ツール概要図

Dockerfileビルド履歴



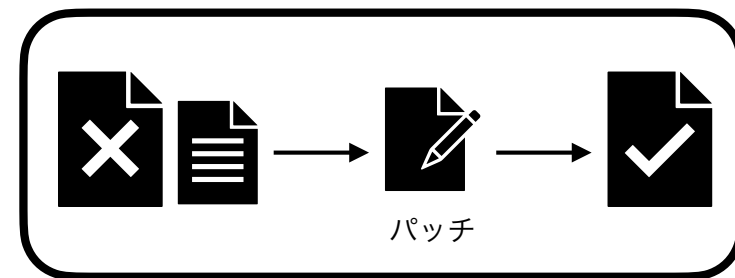
修正データ収集



モデル構築



パッチ生成



ビルドエラーが起こるDockerfileとビルドログから
パッチを生成, 適用して検証

データセット

- ・ GitHub上のDockerfileを含むプロジェクトからCircleCIを使用しているプロジェクトを抽出
- ・ それらプロジェクトからCircleCI上でdocker buildを実行しているプロジェクトを抽出



6603件のプロジェクト

データ収集

- CircleCI APIで過去のDockerfileのビルド履歴データを収集（ビルドログ, コミット情報, 作成日時等）
- コミット情報から対象コミット時のDockerfileを取得
- ビルドログからDockerfileのエラーが起きた行を特定
- ビルドログからエラーを分類（Shipwright）

ビルド結果

成功	25,594 (89.4%)
失敗	2,991 (10.4%)
その他	42 (0.1%)
合計	28,627

エラー分類結果

エラーの内容	件数
pip	1,861 (62.2%)
make	363 (12.1%)
apt	59 (2%)
FROM	52 (1.7%)
go	50 (1.7%)
不明	531 (17.8%)

問題点

- エラーが修正されているがDockerfileに対する変更が行われていないコミットが多数存在していた
 - ▶ Dockerfile以外のファイルを変更することでエラーが修正



- Dockerfileだけでなく修正前後のコミット間で変更があった全てのファイル差分を取得
 - ▶ それらのファイルの修正を行うことでdocker buildのエラーを解消する

これから行うこと

- 収集したデータを分析しエラーごとにどのような修正が行われているか、自動修正が可能かを考察する.
- 自動修正の具体的なアプローチを考える
 - ▶ 修正差分のトークン化, クラスタリング→正規表現
 - ▶ Text-to-Text Transfer Transformer (T5)
 - 学習のために大量のデータが必要
 - Dockerfile以外のファイルも考慮する場合, 1ファイルに対するデータ数が少ない