DiDA: GitHub 上のコード変更と開発履歴に関するデータセット

三島潤平 †　柴藤大介 † †　矢谷浩司 † †

東京大学工学部電子情報工学科 †

東京大学大学院工学系研究科電気系工学専攻 † †

1　はじめに

オープンソースソフトウェア（以下 OSS）の開発は活発に行われている．特に，GitHub[*1]というコードホスティングサービスでは，2017 年時点で，5200 万以上ものプロジェクト[*2]（公開・非公開どちらも含む）が管理されている．そして，GitHub における OSS 開発の過程で生じた膨大な量のデータは，多くの研究者がこれからのソフトウェア開発を発展させるために利用してきた．例えば，第三者による OSS 開発への貢献内容の分析 [5] や，プロジェクト管理に用いられているラベリングの分析 [1, 2]，プロジェクトの類似度計算による類似プロジェクトの検出 [4] などが行われている．また，GitHub の開発履歴データへのアクセスを容易にするため，GHTorrent [3] というデータセットが一般公開されている．今後も，共通のデータセットを用いてソフトウェア開発データの解析をする流れは続いていくだろう．

しかし，開発の過程で生じるデータには，Git によってバージョン管理されているコード変更のデータと GitHub が管理する開発履歴データがあり，両者を統合して大規模に収集したデータセットは存在しない．両方のデータを併せ持つデータセットが存在すれば，これまでは難しかった，コード変更を起点として開発履歴データと結びつけた新しい視点からの解析・発見が生まれる可能性がある．そこで，本研究では，新たなデータ活用の可能性を模索するため，GitHub のコード変更と開発履歴のデータをひとつのデータベースにまとめて収集するシステム（図 1）を構築し，コード変更を使ったデータセット検索ができるアプリケーションを作成した．

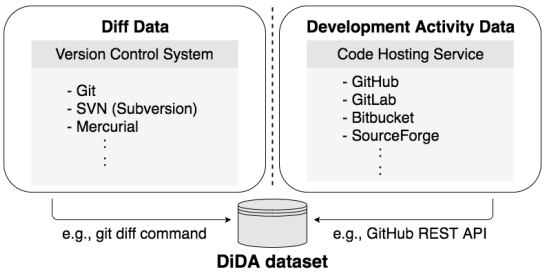


図1: 本研究が収集・統合の対象とするデータの概念図．バージョン管理システムが持つコード変更のデータと，コードホスティングサービスが持つ開発データは別々に入手する必要がある．今回作成した DiDA データセットは，これら 2 種類のデータを収集し統合することにより，より踏み込んだ分析とデータ活用を可能にする．

2　データ収集

本研究が収集・統合の対象としているデータは,図 1に示すように，入手先が 2 つに分かれている．本章では，これら 2 種類のデータがどういったものなのか簡単に解説してから，データ収集システムの説明を行う．

収集するデータについて

本研究が収集の対象とするのは，以下に示す 2 種類のデータである．

- GitHub の開発履歴データ
- GitHub 上で開発されているソースコードのコード差分

GitHub の開発履歴データは，主に GitHub の Issue, Pull Request, Commit を指す．Issue は，プロジェクトの課題管理機能であり，タイトル・本文・ラベルなどからなる．Issue を作成することで，プロジェクトの課題を共有し，コメントによる議論を行うことができる．Pull Request は，コード変更をプロジェクトに統合する前に，他の開発者に変更内容を確認し承認してもらうための機能である．Pull Request はコード変更についての説明や，Issue との紐づけを伴って投稿されるこ

DiDA: A dataset of code diff and development activities on GitHub

Junpei MISHIMA, Daisuke SHIBATO and Koji YATANI
Interactive Intelligent Systems Laboratory,
The University of Tokyo
{junpei, shibato, koji}@iis-lab.org

∗1 GitHub: `https://github.com`

∗2 `https://github.com/features`

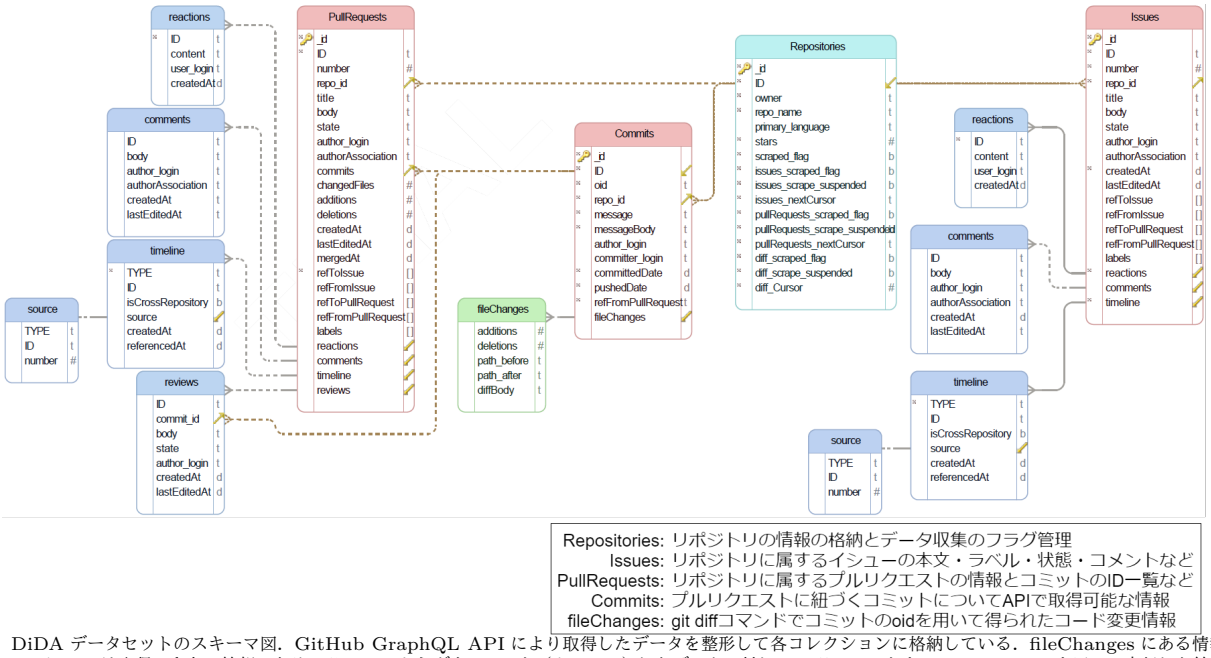


図2: DiDA データセットのスキーマ図．GitHub GraphQL API により取得したデータを整形して各コレクションに格納している．fileChanges にある情報は GitHub API では取得できない情報であり，GitHub からダウンロード（クローン）したデータに対して diff コマンドを commit の oid を用いて実行した結果を格納している．これにより，コード変更のデータと開発データが統合したデータベースとなっている．

とが多く，レビュワー（Pull Request 内のコード変更をチェックする人）は必要があれば修正依頼を出し，問題が無ければ変更内容を merge（マージ）してプロジェクトに統合することができる．Commit は，コード変更を保存する機能で，Pull Request は 1 つまたは複数の Commit から構成されている．

GitHub の開発履歴データは，GitHub が提供する GitHub GraphQL API によって入手することができる．それに対して，GitHub 上で開発されているソースコードのコード差分のデータは Commit に直接対応するものだが，コード差分のデータそのものを API で取得することはできない．コード差分のデータを取得するには，GitHub からプロジェクトのソースコードをダウンロードし，Commit の ID とソースコードのバージョン情報を照らし合わせる必要がある．

自動データ収集システム

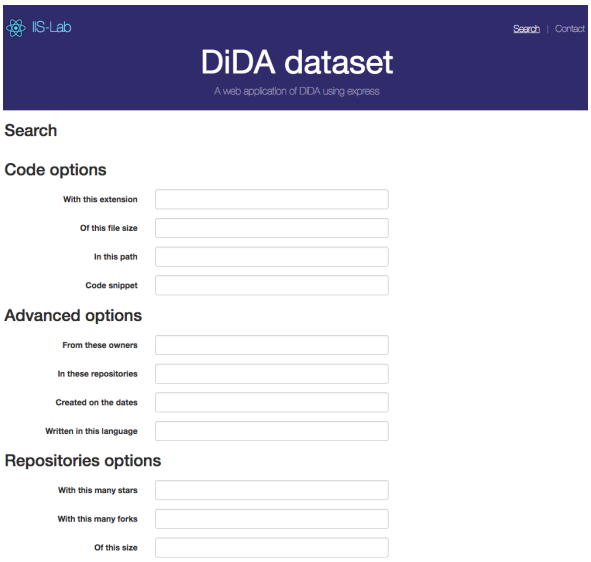
以上のことから，GitHub の開発履歴とコード変更の 2 種類のデータを統合したデータセットを作成するには，3 つの手順が必要になる事がわかる．

1. データ収集対象のプロジェクトのリストを作成する．
2. API を通じて Issue，Pull Request，Commit のデータを取得する．
3. ソースコードをダウンロードして Commit の ID を使ってコード差分のデータを抽出する．

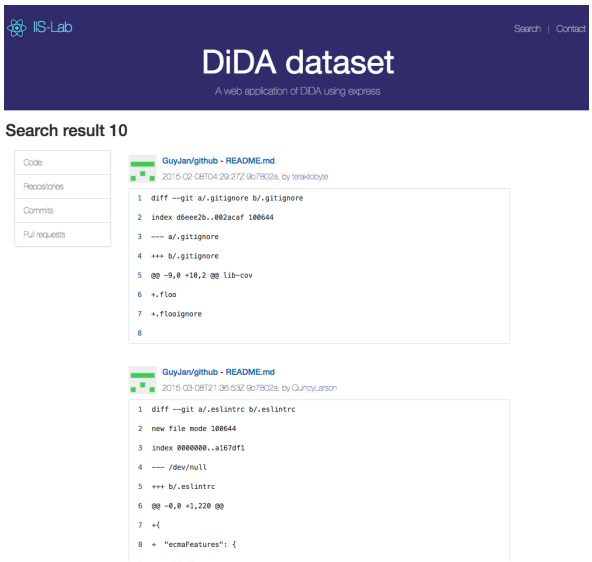
手順 2 . と 3 . は各プロジェクトに対して行う．図 2に収集したデータの格納構造を示す．手順 1 . については，Python を主要プログラミング言語として使用し，かつスター数*3が 11 以上のプロジェクトを検索し，リストを作成した．これは，プロジェクトの検索結果が 1000 件までしか取得できないという GitHub の制限があり，大量のプロジェクトからなるリストを作成するのに時間がかかってしまうからである．Python を指定した理由は，GitHub 上のプロジェクト数が多く，また，Python がネスト構造に決まった数のインデントを使用する言語で，今回収集する断片的なコード変更のデータに対しても分析が比較的容易と考えられるからである．

本データ収集システムは，プロジェクトをリストアップし，データベースにプロジェクトの情報を追加すれば，自動で手順 2. と 3. を反復し，データ収集を行うよう設計した．収集データを格納するデータベースは各所からのアクセスを可能にするため，クラウドプラットフォームである Microsoft Azure Cosmos DB を使用した．また，Cosmos DB において選択可能なラッパ

*3 プロジェクトのお気に入り登録数．スター数が多いほど開発が活発でデータ量が多いとも考えられる．



(a) 検索画面．プロジェクト名やファイル名，コード変更に含まれる文字列などの検索条件を指定して検索することができる．



(b) 検索結果の例．検索条件に当てはまるコード変更が 10 件ずつ表示され，リンクから GitHub の対応するページにアクセスすることができる．

図3: DiDA データセットにアクセスする web インタフェースの画像．React.js で開発している．

ーのうち，API のレスポンスに使用されることの多い JSON 形式のデータ追加が容易で高速な検索が可能な MongoDB を採用した．データ収集スクリプトの実行は，Microsoft Azure の Linux 仮想マシン（Standard DS3 v2 Promo）上で行った．使用した仮想マシンは cpu 数が 4，メモリが14 GBである．GitHub GraphQL API へのアクセスには 1 時間あたり 5000 ポイント（ポイント数は GitHub が独自に算出）までという制限があるが，データ収集スクリプトを 4 並列で実行したことにより，アクセス限界に近いペースでデータ収集を行うことができた．アクセス可能ポイント数の残量を監視し，残り 100 ポイント以下になった場合はポイントが回復するまでデータ収集を中断するようにした．

3　収集したデータの概要

本研究で，収集を行ったデータセットを DiDA（Diff & Development Activity）データセットと呼ぶことにした．収集対象のプロジェクトは計 53573 件あったが，表 1に示すように，Issue，Pull Request，Diff 全てのデータを収集できたプロジェクトは全体の60.2 %にあたる 32263 件であった．Diff のデータのみが存在しないプロジェクトは，全体の35.8 %にあたる 19194 件であるが，これは Commit が存在しないか Pull Request に紐づいておらず，コード変更のデータの取得ができなかったプロジェクトであると考えられる．その他の 2116 件のプロジェクトについては何らかの理由でデータ収集が完了していないが，原因は不明である．

データ収集は約 2 週間に渡って行われ，表 2に示すように，約118 GBのデータを収集することができた．内訳として，Issue は約 287 万件，Pull Request は約 239 万件，Commit は 741 万件集まった．

4　コード変更の検索

本研究で作成した DiDA データセットは，コード変更を検索ワードとする新たな情報入手手段としても使用

表1: プロジェクトのデータ収集完了状況．

	プロジェクト数	割合
開発履歴＋コード変更	32,263	60.22 %
開発履歴のみ	19,194	35.83 %
その他欠損情報あり	2,116	3.95 %
計	53,573	

表2: 各種データの収集数とデータ容量．

コレクション ID	データ数	データ容量
Repositories	53,573	96.86 MB
Issues	2,871,559	20.65 GB
PullRequests	2,385,669	18.50 GB
Commits	7,411,389	78.98 GB

することができる．例えば，大規模なソフトウェア開発に不慣れな開発者が，あるコード変更に伴う他のファイルの変更例を調べたいときなどに，コード変更を使った検索は役に立つ．そこで，我々は DiDA データセットを活用する 1 つの方法を提供するため，コード変更を検索ワードとして DiDA データセットを検索できるアプリケーションを作成した．実装したアプリケーションは，図 3のように，ブラウザの web ページを通してアクセスできる．図 3aのように，各検索ボックスにリポジトリ名やファイル名，コード変更の一部を検索条件として入力することでデータの検索が実行できる．検索の実行結果は図 3bのようになっており，検索条件に合致するコード変更のデータが 10 件ずつ表示される．コード変更やコード変更に関連する Pull Request へのリンクが貼られており，クリックすることで，GitHub の対応するページにアクセスすることができる．本インタフェースの表示部分は，ページ内の再描画が円滑に行えるように React.js 環境を使用して開発された．検索フォームに入力された文字列は，DiDA データセットのデータベースに対するクエリに整形される．また検索結果は 10 件ずつ表示され，「次へ」ボタンをクリックすることにより，次の 10 件に対する検索結果が表示される．

5　おわりに

本稿では，GitHub のコード変更と開発履歴データを統合した DiDA データセットの作成を行い，web インタフェースを通じたアクセスを可能にした．今後は以下の課題に取り組む予定である．

- データセットの拡充と，クエリの処理速度向上
- DiDA データセットを使ったアプリケーションの検討

参考文献

[1] Bissyandé, T. F., Lo, D., Jiang, L., Rébeillère, L., Klein, J., and Traon, Y. L.: Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub, Proc. *ISSRE '13*, pp.188–197, IEEE (2013).

[2] Cabot, J., Cánovas Izquierdo, J. L., Cosentino, V., and Rolandi, B.: Exploring the use of labels to categorize issues in Open-Source Software projects, Proc. *SANER '15*, pp.550–554, IEEE (2015).

[3] Georgios, G., Bogdan, V. Alexander, S., and Andy, Z.: Lean GHTorrent: GitHu Data on Demand, Proc. *MSR '14*, pp.384–387, ACM (2014)

[4] Mohamed, G., Abdelmalek, A., and Reda, M. H.: Recommending Relevant Open Source Projects on GitHub Using a Collaborative-Filtering Technique. Int. J. Open Source Softw. Process. 6, 1, pp.1–16 (2015)

[5] Pinto, G., Steinmacher, I., and Gerosa, M. A.: More Common Than You Think: An In-depth Study of Casual Contributors, Proc. *SANER '16*, pp.112–123, IEEE (2016)